The CWEB System of Structured Documentation

(Version 3.64 — February 2002)

Donald E. Knuth and Silvio Levy

Internet page `http://www-cs-faculty.stanford.edu/~knuth/cweb.html` contains current info about CWEB and related topics.

El CWEB Sistema de Documentación Estructurada

(Versión 3.64 — Febrero 2002)

Donald E. Knuth y Silvio Levy

La Página de Internet `http://www-cs-faculty.stanford.edu/~knuth/cweb.html` contiene información actual de CWEB al respecto y tópicos relacionados.

El CWEB Sistema de Documentación Estructurada

Donald E. Knuth y Silvio Levy

Este documento describe una versión del Sistema WEB de Don Knuth, adaptado a C por Silvio Levy. Desde su creación en 1987, CWEB ha sido revisado y mejorado de varias formas, por Knuth y Levy. Ahora creemos que su evolución está cerca del fin; sin embargo, reportes de errores, sugerencias y comentarios son todavía bienvenidos y deberán ser enviados a Levy (levy@math.berkeley.edu).

Los Lectores que están familiares con el memorándum de Knuth "El CWEB Sistema de Documentación Estructurada" (*"The* WEB *System of Structured Documentation"*) pueden leer superficialmente este material, porque CWEB y WEB comparten la misma filosofía y (esencialmente) la misma sintaxis. En algunos temas CWEB es una simplificación de WEB: por ejemplo, CWEB no necesita las características de WEB para definición de macro y manejo de cadena de caracteres (*string*), porque C y su preprocesador ya manejan macros y cadena de caracteres. De la misma forma, las convenciones WEB de denotar constantes octales y hexadecimales por @'77 y @"3f son reemplazadas por las convenciones de C's 077 y 0x3f. Todas las otras características de WEB han sido retenidas y han sido agregadas nuevas características.

Se agrade a todos quienes contribuyeron y criticaron al desarrollo de CWEB, especialmente a Steve Avery, Nelson Beebe, Hans-Hermann Bode, Klaus Guntermann, Norman Ramsey, Joachim Schnitter y Saroj Mahapatra, quienes contribuyeron con código, y a Cameron Smith, quien hizo muchas sugerencias mejorando el manual. Ramsey ha hecho accesible la programación literaria (*literate programming*) a usuarios que incluso usan otros lenguajes por medio de su sistema SPIDER [véase *Communications of the ACM* **32** (1989), páginas 1051 a 1055]. El libro *Literate Programming* por Knuth (1992) contiene una bibliografía exhaustiva de trabajos iniciales relacionados. Bode, Schnitter y Mahapatra adaptaron CWEB así que funciona para C++ también; por lo tanto, en el texto de abajo puede leer C++ en vez de C si lo desea.

Introducción

La filosofía detrás de CWEB es que los programadores que deseen proveer la mejor documentación posible a sus programas necesitan dos cosas simultáneamente: por una parte un lenguaje como TEX para el formateo; y por el otro un lenguaje como C para la programación. Ningún tipo de lenguaje puede proveer la mejor documentación por sí mismo. Pero cuando ambos son apropiadamente combinados, obtenemos un sistema que es mucho más útil que cualquiera por separado.

La estructura de un programa de software podría ser pensado como una "*web*" ("telaraña/tejido") que estará hecha por muchas piezas interconectadas. Para documentar tal programa, queremos explicar cada parte de la *web* y cómo se relaciona con sus vecinos. Las herramientas tipográficas provistas por TEX nos dan una oportunidad para explicar la estructura local de cada parte al hacer esa estructura visible, y las herramientas de programación provistas por C nos hace posible especificar los algoritmos formalmente y sin ambigüedades. Combinando los dos, podemos desarrollar un estilo de programación que maximiza nuestra capacidad para percibir la estructura de un software complejo, y al mismo tiempo los programas documentados pueden ser mecánicamente traducidos a un sistema de software funcionando que es correlativo a la documentación.

El sistema CWEB consiste de dos programas llamados CWEAVE y CTANGLE. Cuando se escribe un programa CWEB el usuario mantiene el código C y la documentación en el mismo archivo, llamado el archivo CWEB y generalmente denominado something.w. El comando '`cweave something`' crea un archivo de salida something.tex, el cual puede ser pasado a TEX, entregando una versión de "impresión bonita" ("*pretty printed*") de something.w que maneja correctamente detalles tipográficos tal como el diseño de página y el uso de sangría (*indent*), cursiva, negrita y símbolos matemáticos. La composición de salida también incluye información extensiva de índices cruzados (*cross-index*) que es contenido automáticamente.

Igualmente, si tu corres el comando '`ctangle something`' obtendrás un archivo C something.c, el cual puede ser compilado logrando un ejecutable.

Además de proveer una herramienta para documentar, CWEB mejora el lenguaje C proveyendo la capacidad de permutar partes del texto del programa, así un sistema grande puede ser entendido completamente en términos de secciones pequeñas y sus interrelaciones locales. El programa CTANGLE (*to tangle*: enredar) es llamado así porque toma una *web* dada y mueves las secciones de esa estructura al orden requerido por C; la ventaja de programar en CWEB es que los algoritmos pueden ser expresados de la forma"*untangled*"

("desenredadas"), con cada sección explicada separadamente. El programa CWEAVE es llamado así porque toma una *web* y entrelaza porciones del TEX y C contenidas en cada sección, entonces junta el tejido completo dentro de un documento estructurado. (¿entiendes? ¡Ah!) Quizás hay alguna conexión profunda aquí con el hecho que la palabra "*weave*" (tejer) en alemán es "*webe*", y el imperativo correspondiente en latín es ¡"*texe*"!

Un usuario de CWEB debería estar familiarizado con el lenguaje de programación C. Una pequeña cantidad de conocimiento con TEX también es deseable, pero de hecho puede ser aprendido a medida que se usa CWEB, el hecho que el texto simple pueda ser compuesto en TEX virtualmente sin el conocimiento del lenguaje. Para alguien familiar con C y TEX la cantidad de esfuerzo necesario para aprender los comandos de CWEB es pequeño.

Visión General

Dos tipos de material van dentro de los archivos CWEB: el texto TEX y el texto C. Un programador escribiendo en CWEB deberá pensar en ambas documentaciones y del programa C que está siendo creado; i.e.:, el programador deberá estar instintivamente consciente de las diferentes acciones que CWEAVE y CTANGLE desempeñarán en el archivo CWEB. El texto TEX es esencialmente copiado sin cambios por CWEAVE y es totalmente borrado por CTANGLE; El texto TEX es "documentación pura." El texto C, por otro lado, es formateado por CWEAVE y es reorganizado por CTANGLE, de acuerdo a las reglas que serán más claras después. Por ahora el punto importante es tener claro que hay dos tipos de textos. Escribir programas CWEB es similar a escribir documentos TEX pero en un "modo C" adicional. que es agregado al modo horizontal de TEX, modo vertical y modo matemático.

Un archivo CWEB es construido desde unidades llamadas *secciones* (*sections*) que están más o menos autocontenidas. Cada sección tiene tres partes:

- Una parte de TEX, conteniendo material explicativo sobre que acontece en la sección.
- Una parte intermedia, conteniendo definiciones macro que sirven como abreviaciones para construcciones de C que serían menos entendibles si fuesen escritas completas cada vez. Ellas son transformadas por CTANGLE en definiciones macro de preprocesador.
- Una parte de C, conteniendo un pedazo del programa que CTANGLE producirá. Este código C deberá idealmente ser alrededor de una docena de líneas, así es fácilmente entendible como una unidad, de la misma forma su estructura es inmediatamente percibida.

Las tres partes de cada sección deberán aparecer en este orden; i.e.:, El comentario TEX deberá venir primero, después la parte del medio, y finalmente, el código C. Cualesquiera de las partes podrá estar vacía.

Una sección comienza con alguno de los símbolos '`@␣`' o '`@*`', donde '␣' denota un espacio en blanco. Una sección termina en el principio de la próxima sección (i.e., al siguiente '`@␣`' o '`@*`'), o al final del archivo, cualquiera que venga primero. El archivo CWEB podrá también contener material que no es parte de ninguna sección, concretamente el texto (si hay alguno) que está justo antes de la primera sección. A tal texto se le nombra como estar "en el limbo"; Es ignorado por CTANGLE y copiado textualmente por CWEAVE, así su función es proveer otras instrucciones adicionales que podrían querer ponerse en la salida de TEX. De hecho, es habitual comenzar un archivo CWEB con código TEX en el limbo que cargue fuentes especiales, defina marcos especiales, cambie los tamaños de las páginas y/o produzca un página de título.

Las secciones son enumeradas consecutivamente, empezando con 1. Estos números aparecen al principio de cada sección de los documentos de salida de TEX hecha por CWEAVE y ellos aparecerán como comentarios en paréntesis al principio y final del código generado por esa sección en la salida del programa en C hecha por CTANGLE.

Nombres de las Secciones

Afortunadamente, nunca mencionas estos números cuando estás escribiendo en CWEB. Sólo dices '`@␣`' o '`@*`' al principio de cada sección nueva y los números son suministrados automáticamente por CWEAVE y CTANGLE. De esta forma, una sección tiene un *nombre* en vez de un número; su nombre está especificado al escribir '`@<`' seguido por texto TEX seguido por '`@>`'. Cuando CWEAVE libere por su salida un nombre de sección (*section name*), reemplaza '`@<`' y '`@>`' por paréntesis angulares e inserta el número de la sección a un tipo de letra pequeño. Así, cuando leas la salida de CWEAVE es fácil ubicar cualquier sección que sea referida en otra.

Para propósitos explicativos, un nombre de sección deberá ser una buena descripción de los contenidos de ella; i.e., deberá tomar una postura para la abstracción representada por la sección. Entonces ella puede ser

"insertada dentro de" (*"plugged into"*) una o más secciones de tal forma que detalles no importantes de sus trabajos internos sean suprimidos. Un nombre de sección, por lo tanto, debería ser lo suficientemente largo para traspasar el significado necesario.

Desafortunadamente, es laborioso escribir tales nombres largos una y otra vez, y es difícil también especificar un nombre largo dos veces en la misma forma exacta que CWEAVE y CTANGLE podrán correlacionar los nombres a sus secciones. Para mejorar esta situación, CWEAVE y CTANGLE te deja abreviar un nombre de sección, mientras tanto que el nombre completo aparezca en algún lado del archivo CWEB; puedes simplemente escribir '`@<`α`...@>`', donde α es cualquiera cadena de caracteres que es prefijo de exactamente un nombre de sección que aparece en el archivo. Por ejemplo, '`@<Clear the arrays@>`' puede ser abreviado como '`@<Clear...@>`' Si no otro nombre de sección comienza con las cinco letras '`Clear`'. En cualquier otro lugar podrías usar las abreviaciones '`@<Clear t...@>`', etc.

De otro modo los nombres de sección deberán correlacionar carácter por carácter, excepto esos caracteres consecutivos de espacios en blanco (espacios, tabulaciones (*tab marks*), nuevas líneas (*newlines*) y/o alimentación de línea (*form feeds*) son tratadas como un equivalente de sólo un espacio y tales espacios son borrados al principio y final del nombre. Así, '`@< Clear  the arrays @>`' también correlacionará el nombre en el ejemplo anterior. Espacios seguidos de puntos suspensivos en abreviaciones son ignorados también, pero no los que están antes, de esta manera '`@<Clear t ...@>`' no se correlacionará con '`@<Clear the arrays@>`'.

Que CTANGLE Hace

Hemos dicho que una sección comienza con '`@␣`' o '`@*`', pero no hemos dicho como se divide dentro de la parte de TEX, una parte intermedia y una parte C. La parte intermedia empieza con la primera aparición de '`@d`' o '`@f`' en la sección, y la parte de C empieza con la primera aparición de '`@c`' o '`@<section name@>=`'. En este último caso estás diciendo, de hecho, que el nombre de la sección significa el texto C que continua. Otra posibilidad, si la parte C empieza con '`@c`' en vez del nombre de la sección, la sección actual se dice ser *sin nombre* (*unnamed*).

El constructor '`@<section name@>`' puede aparecer las veces que sea en la parte de C de una sección: Apariciones subsiguientes indican que una sección con-nombre está siendo "usada" más que "definida." En otras palabras, el código C para la sección con-nombre (supuestamente definidas en otro lugar) deberán unirse en este punto en el programa C. De hecho, la idea principal de CTANGLE es hacer un programa C de secciones individuales, con y sin nombres. La forma exacta que esto es hecho es: Primero todas las definiciones de macro indicadas por '`@d`' son transformadas en definiciones de macro del preprocesador C y copiadas al principio. Entonces las partes de C de secciones sin nombre son copiadas, en orden; esto constituye la aproximación de primer-orden al texto del programa. (Deberá haber al menos una sección sin nombre, de otra manera no habrá programa.) Después todas los nombres de secciones que aparecen en la aproximación de primer orden son reemplazadas por las partes de C de las secciones correspondientes y este proceso de substitución continua hasta que no quede ningún nombre de sección. Todos los comentarios son removidos, porque el programa C es intencionalmente para los ojos de un compilador C.

Si el mismo nombre ha sido dado a mas de una sección, el texto C para ese nombre es obtenido a través de poner juntos todas las partes de C en las secciones correspondientes. Esta característica es útil, por ejemplo, en una sección llamada 'Variables globales', como una puede entonces declara variables globales en cualquier sección donde esas variables son introducidas. Cuando varias secciones tienen el mismo nombre, CWEAVE asigna el número de la primera sección como el número correspondiente a ese nombre, e inserta una nota al final de esa sección diciendo al lector 'Mirar también las secciones una-y-otra'; este pie de página entrega los números de todas las otras secciones con el mismo nombre tal cual el presente. El texto C correspondiente a una sección es usualmente formateado por CWEAVE de esta forma la salida tiene un signo de igualdad en lugar del signo igual en el archivo CWEB; i.e., la salida dice '$\langle$ nombre de sección $\rangle \equiv$ texto C'. Sin embargo, en los casos de las apariciones segunda y subsiguientes de una sección con el mismo nombre, este signo '$\equiv$' es reemplazado por '$+\equiv$', como una indicación que el siguiente texto C está agregado al texto C de otra sección.

Como CTANGLE entra y sale de secciones, inserta comandos `#line` del preprocesador dentro del archivo de salida C. Esto significa que cuando el compilador te da mensajes de error o cuando depuras tu programa, los mensajes se refieren a los números de línea en el archivo CWEB y no a las del archivo C. En la mayoría de los casos de esta manera te podrás olvidar sobre el archivo C totalmente.

Que CWEAVE Hace

La idea general de CWEAVE es hacer un archivo .tex desde el archivo CWEB de la siguiente manera: La primera línea del archivo .tex le dice a TEX que ingrese un archivo con macros que definan las convenciones de documentación de CWEB. Las líneas siguientes del archivo serán copiadas desde el texto TEX que sea que está en el limbo antes de la primera sección. Después viene la salida de cada sección sucesivamente, posiblemente intercalado entre marcas de fin de página (*end-of-page*). Finalmente, CWEAVE generará un índice de referencia cruzada (*remisión*) que cataloga cada número de sección en cual cada identificador C aparece y también generará una lista en orden alfabético de los nombres de las secciones, como también una tabla de los contenidos que muestra la página y números de la sección de cada sección "estrellada" ("*starred*", marcada con un asterisco).

¿Qué es una sección "estrellada", preguntas? Una sección que comienza con '@*' en vez de '@␣' es levemente especial en que denota un nuevo grupo mayor de secciones. El '@*' debería ser seguido por el título de este grupo, seguido por un punto. Tales secciones siempre comenzarán en una página nueva en la salida de TEX y el título del grupo aparecerá como un titular continuo en todas las páginas subsiguientes hasta la próxima sección estrellada. El título también aparecerá en la tabla de contenidos y en negrita al principio de su sección. Advertencia: No uses secuencias de control (*control sequences*) de TEX en tales títulos, a menos que sepas que las macros cwebmac harán lo correcto con ellos. La razón es que estos títulos son convertidos a letras mayúsculas cuando aparecen como titulares continuos y son convertidos a negritas cuando aparecen al principio de sus secciones y son también escritos a un archivo de tabla de contenidos usado para almacenamiento temporal mientras TEX está trabajando; cualesquiera secuencias de control se use deberán ser significativas en cualquier de estos tres modos.

La salida TEX producida por CWEAVE para cada sección consiste en lo siguiente: Primero viene el número de la sección (p. ej., '\M123.' al principio de la sección 123, excepto que '\N' aparezca en lugar de '\M' al principio de una sección estrellada). Luego viene la parte de TEX de la sección, copiada casi textualmente excepto como está descrito abajo. Después viene la parte intermedia y la parte C, formateada así que habrá un pequeño espacio extra entre ellos si ambos no son vacíos. Las partes intermedia y de C son obtenidas al insertar un montón de macros TEX de apariencia divertida dentro del programa C; estas macros administran los detalles tipográficos acerca las fuentes y espacio matemático apropiado, tal como salto de líneas y sangrías.

Código C en Texto TEX y Viceversa

Cuando estás escribiendo texto TEX, probablemente querrás hacer referencias frecuentes a variables y otras incógnitas en tu código C y querrás que esas variables tengan el mismo trato tipográfico cuando ellas aparezcan en tu texto y en tu programa. Por lo tanto, el lenguaje CWEB te permite obtener los efectos de C editados dentro de texto TEX, si pones marcas '|' antes y después del material C. Por ejemplo, supón quieres decir algo como esto:

> If *pa* is declared as '**int** $*pa$', the assignment $pa = \&a[0]$ makes *pa* point to the zeroth element of *a*.

El texto TEX lucirá como esto en tu archivo CWEB:

```
If |pa| is declared as '|int *pa|', the
assignment |pa=&a[0]| makes |pa| point to the zeroth element of |a|.
```

y CWEAVE traduce esto a algo que estarás alegre de no tener que escribirlo:

```
If \\{pa} is declared as '\&{int} ${}{*}\\{pa}$',
the assignment $\\{pa}\K{\AND}\|a[\T{0}]$
makes \\{pa} point to the zeroth element of \|a.
```

Por casualidad, el índice de referencia-cruzada que CWEAVE hará, en la presencia de un comentario como este, incluirá el número de sección actual como una de las entradas del índice para *pa*, aunque *pa* podría no aparecer en la parte de C de esta sección. De este modo, el índice cubre referencias a identificadores en los comentarios explicativos como también en el mismo programa; pronto aprenderás apreciar esta característica. Sin embargo, los identificadores **int** y *a* no serán indexados, porque CWEAVE no hace entradas al índice de palabras reservadas o identificadores de letras-simples. Tales identificadores son tan ubicables que no tendria sentido mencionar en cada lugar que ocurra.

Aunque una sección empieza con texto TEX y termina con texto C, hemos notado que la línea de división no es nítida, ya que el texto C puede ser incluido en el texto TEX si está encerrado en '|...|'. A la inversa,

el texto TEX aparece frecuentemente con texto C, porque todo lo que esté en los comentarios (i.e., entre `/*` y `*/`, o después de `//`) es tratado como texto TEX. Asimismo, el texto de un nombre de sección consiste de texto TEX, pero el constructor `@<section name@>` como un todo es esperado ser encontrado en el texto C; así, uno usualmente va y viene de una forma natural entre los ambientes de C y TEX, tal como en estos ejemplos:

```
if (x==0) @<Empty the |buffer| array@>
... using the algorithm in |@<Empty the |buffer| array@>|.
```

El primero de estos pasajes será encontrado en la parte C de una sección, dentro de la cual el código de la sección llamada "Empty the *buffer* array" está siendo unido. El segundo pasaje será encontrado en la parte de TEX de la sección, y la sección con-nombre está siendo "citada" más que siendo definida o usada. (Nota '`|...|`' rodeando el nombre de la sección en este caso.)

Macros

El código de control (*control code*) `@d` seguido por

identifier C text or by *identifier*`(`$par_1, \ldots, par_n$`)` C text

(donde no hay espacios en blanco entre el *identificador* y los paréntesis y el segundo caso) es transformado por `CTANGLE` a un comando de preprocesador, empezando con `#define`, cual es imprimido en la parte superior del archivo de salida de C tal como ya fue explicado.

Una definición macro '`@d`' puede ir en varias líneas y en las nuevas líneas no tienen que ser protegidas por barra diagonal invertida (*backslash*), ya que `CTANGLE` inserta las barras diagonales invertidas. Si por alguna razón necesitas definir un comando `#define` en un lugar específico en tu archivo C, puedes tratarlo como código C, en vez de una macro `CWEB`; pero ahí, tienes que proteger las nuevas líneas.

Cadena de caracteres y constantes

Si quieres que una cadena de caracteres aparezca en el archivo C, delimitado por pares de marcas de `'` o `"` como siempre, puedes escribirlos exactamente igual en el archivo `CWEB`, excepto por el carácter '`@`' que deberá ser escrito '`@@`' (se transforma en un código de control, el único que puede aparecer en una cadena de caracteres; véase abajo). Las cadena de caracteres deberán terminar en la misma línea que comienzan, a menos que haya una barra diagonal invertida al final de las líneas.

TEX y C tienen diferentes formas de referirse a constantes octales y hexadecimales, porque TEX está orientado a la escritura técnica por mientras C está orientado al proceso de computadoras. En TEX para hacer una constante octal o hexadecimal hay que poner de prefijo `'` ó `"`, respectivamente; en C la constante deberá estar precedida por `0` o `0x`. En `CWEB` parece razonable dejar cada convención con su respectivo contexto; de esta manera en texto C se obtiene 40_8 al escribir '`040`', al cual `CTANGLE` copia fielmente en los archivos C (para beneficio del compilador) y donde `CWEAVE` lo imprime como $^\circ 40$. Del mismo modo, `CWEAVE` imprime la constante C hexadecimal '`0x20`' como $^\#20$. El uso de fuentes cursivas para dígitos octales y fuente de máquina de escribir (*typewriter*) para dígitos hexadecimales hacen el significado de tales constantes más claro en el documento. Para mantener consistencia, entonces deberás escribir '`|040|`' o '`|0x20|`' en la parte de TEX de la sección.

Códigos de Control

Un *código de control* (*control code*) `CWEB` es una combinación de dos caracteres de la cual el primer es '`@`'. Ya hemos visto el significado de varios códigos de control; es tiempo de listarlos más metódicamente.

En la siguiente lista, las letras en corchetes después del código de control indica los contextos que el código es permitido. L indica que el código es permitido en el limbo; T (para TEX), M (para intermedio) y C (para C) quiere decir que el código es permitido en cada una de las tres partes de la sección, a nivel superior, eso es, fuera de los constructores como '`|...|`' y nombres de sección. Una flecha $\rightarrow$ significa que el código de control termina la parte actual del archivo `CWEB` e inaugura la parte indicada por la letra después de la flecha. Así $[LTMC \rightarrow T]$ al lado de `@␣` indica que este código de control puede ocurrir en el limbo o en cualquiera de estas tres partes de la sección y ahí comienza la (posiblemente vacía) parte de TEX de la siguiente sección.

Las otras abreviaciones pueden ocurrir en estos corchetes: La letra r significa *contexto restringido* (*restricted context*), es decir, material dentro de comentarios C, nombres de sección, cadena de caracteres C y

textos de control (*control texts*) (definidos abajo); la letra *c* significa *contexto* C *interior* (*inner* C *context*), es decir, material C dentro de '`|...|`' (incluyendo los comentarios internos de '`|...|`, pero no esos que están en otros contextos restringidos). Un asterisco $*$ seguido de corchetes significa que el contexto de este código de control correlativo a `@>` está restringido.

Códigos de control con letras que no son sensibles a las mayúsculas/minúsculas (*case-sensitive*); de esta forma `@d` y `@D` son equivalentes. Solamente las versiones en minúsculas son mencionadas específicamente abajo.

`@@` [$LTMCrc$] Un `@` doble denota el carácter '`@`'. Este es el único código de control que es legal en todos lados. Fíjate que deberás usar esta convención si estás colocando un correo electrónico en un archivo `CWEB` (p. ej., `levy@@math.berkeley.edu`).

Aquí están los códigos que introducen la parte TEX de una sección.

`@␣` [$LTMC \to T$] Esto denota el principio de una nueva sección no estrellada (*unstarred*). Una tabulación o alimentación de página o carácter de fin de línea (*end-of-line*) es equivalente a un espacio después de un signo `@` (y en la mayoría de los casos).

`@*` [$LTMC \to T$] Esto denota el principio de una sección nueva estrellada, i.e., una sección que empieza con un grupo mayor nuevo. El título del grupo nuevo deberá aparecer después de `@*`, seguido de un punto. Tal como fue explicado arriba, secuencias de control de TEX deberán ser evitadas in tales títulos a menos que sean muy simples. Cuando `CWEAVE` y `CTANGLE` leen un `@*`, imprimen un asterisco en el terminal seguido por el número de sección actual, de esta manera el usuario puede ver algunas indicaciones de progreso. La primera sección debería ser estrellada.

Puedes especificar la "profundidad" ("*depth*") de una sección estrellada al escribir `*` o un número decimal después de `@*`; esto indica la clasificación relativa de grupo actual de secciones en la jerarquía del programa. Se dice que tienen profundidad -1 porciones del programa de capa superior (*top-level*), dadas por `@**`, con sus nombres compuestos en negrita en la tabla de contenidos. si no la profundidad es un número no negativo, el cual determina la cantidad de sangría en las páginas de contenido. Tal sangría ayuda a clarificar la estructura de una programa largo. La profundidad es asumida en 0 si no es explícitamente especificada; cuando tu programa es corto, deberás dejar todas las profundidades en cero. Una sección estrellada siempre comienza con una página nueva de salida, a menos que la profundidad sea mayor que 1.

La parte intermedia de cada sección consiste de algún número de definiciones de macro que (empiecen con `@d`) y definiciones de formatos (empiecen con `@f` o `@s`), entre mezclados en cualquier orden.

`@d` [$TM \to M$] Definiciones Macro comienzan con `@d`, seguido de un identificador y parámetros opcionales y texto C tal como ya fue explicado.

`@f` [$TM \to M$] Definiciones de formato comienzan con `@f`; causan que `CWEAVE` trate los identificadores de una manera especial cuando aparecen en texto C. La forma general de una definición de formato es '`@f` *l r*', seguido de un comentario opcional entre `/*` y `*/`, donde *l* y *r* son identificados; posteriormente `CWEAVE` tratará el identificador *l* tal como trata a *r* actualmente. Esta característica permite a un programador `CWEB` inventar nuevas palabras reservados y/o des-reservar algunas de los identificadores reservados de C. Por ejemplo, las palabras comunes 'error' y 'line' se le ha dado un significado especial en el preprocesador de C, de manera que `CWEAVE` es configurado para formatearlos especialmente; si quieres una variable llamada *error* o *line*, deberá decir

```
@f error normal        @f line normal
```

en algún lado de tu programa.

Si *r* es el identificador especial '*TeX*', el identificador *l* será formateado como un control de secuencia TEX; por ejemplo, '`@f foo TeX`' en el archivo `CWEB` causará que el identificador *foo* sea la salida como `\foo` por `CWEAVE`. El programador deberá definir `\foo` para tener el formato personalizado deseado, asumiendo el modo de matemáticas de TEX. (Cada carácter subrayado es convertido a `x` cuando se hace la secuencia de control de TEX, y cada signo de dólar es convertido a `X`; de este modo *foo_bar* se convierte en `\fooxbar`. Otros caracteres, incluyendo dígitos, son dejados sin traducir, así TEX los considerará como parámetros de macro, no como parte de la misma secuencia de control. Por ejemplo,

```
\def\x#1{x_{#1}} @f x1 TeX @f x2 TeX
```

formateará `x1` y `x2` no como *x1* y *x2* pero si como x_1 y x_2.)

Si r es el identificador especial '*make_pair*', el identificador l será tratado como una plantilla (*template*) de función de C++. Por ejemplo, después `@f convert make_pair` uno puede decir '`convert<int>(2.5)`' sin tener a `<` and `>` como un mal entendido de los signos menor-que y mayor-que.

`CWEAVE` conoce que los identificadores definidos con un **typedef** deberán convertirse en palabras reservadas; así ne necesitas formatear definiciones muy seguido.

`@s` $[TM \to M;\ L]$ igual que `@f`, pero `CWEAVE` no muestra la definición de formato en la salida y el comentario opcional C no está permitido. Esto es usado generalmente en los archivos `@i`.

A continuación se muestra los códigos que determinan la parte de C de una sección.

`@c` `@p` $[TM \to C]$ La parte C de una sección sin-nombre empieza con `@c` (o con `@p` para "programa"; ambos código de control hacen lo mismo). Esto causa que `CTANGLE` agregue los siguientes códigos C al texto del programa de primer-orden, como fue explicado en la página 3. Nota que `CWEAVE` no imprime un '`@c`' en la salida TEX, de esta forma si estas creando un archivo `CWEB` basado en una documentación `CWEB` de TEX-impresa debes recordar insertar `@c` en los lugares apropiados de las secciones sin-nombre.

`@<` $[TM \to C;\ C;\ c]$ $*$ Este código de control introduce un nombre de sección (o prefijo ambiguo, tal como fue discutido arriba), el cual consiste en texto TEX y se extiende hasta encontrar `@>`. El constructor `@<...@>` es desde un punto de vista conceptual un elemento C. Este comportamiento es diferente dependiendo del contexto:

Un `@<` apareciendo en los contextos T y M adjunta al siguiente nombre de sección con la sección actual e inaugura la parte C de ella. El `@>` que cierra deberá ser seguido por `=` o `+=`.

En contexto C, `@<` indica que la sección con-nombre está siendo usada, su definición C es unida por `CTANGLE`, como fue explicada en la página 3. Como una medida de detección-de-errores, `CTANGLE` y `CWEAVE` reclaman si ese nombre de sección es seguido por `=`, principalmente esto es el significado de la definición de una nueva sección y deberá ser precedida por `@␣`. Si realmente quieres decir $\langle\,\text{foo}\,\rangle = bar$, donde $\langle\,\text{foo}\,\rangle$ está siendo usada y no definida, pon una línea nueva antes del `=`.

Por último, en el contexto interno C (eso es, dentro de '`|...|`' en la parte TEX de una sección o en un comentario), `@<...@>` significa que la sección con-nombre está siendo citada. Una incidencia así es ignorada por `CTANGLE`. Nota que incluso aquí pensamos que el nombre de sección es un elemento C, por consiguiente `|...|`.

`@(` $[TM \to C;\ C;\ c]$ $*$ Un nombre de sección puede empezar con `@(`. Todo funciona bien para `@<`, excepto que el código C de la sección con-nombre `@(foo@>` es escrito por `CTANGLE` al archivo `foo`. De esta manera puedes obtener múltiples-archivos de salida desde un simple archivo `CWEB`. (Las definiciones `@d` son colocadas en la salida en tales archivos, sólo al archivo maestro `.c`.) Un uso de esta característica es producir archivos de cabecera para otros módulos de programa que serán cargados con el presente. Otro uso es producir una rutina de test que vaya con tu programa. Al mantener las fuentes para un programa y su cabecera y rutina de test juntas, es más probable que mantendrás a las tres consistentes entre ellas. Nota que la salida de una sección con-nombre puede ser incorporada en varios archivos de salida diferentes, porque puedes mencionar `@<foo@>` en `@(bar1@>` y `@(bar2@>`.

`@h` $[Cc]$ Causa a `CTANGLE` insertar en el lugar actual las declaraciones `#define` de las partes intermedias de todas las secciones y *no* escribirlas al principio del archivo C. Es útil cuando quieres que las definiciones de macro vengan después de los archivos incluidos (*include files*). (Ignorado por `CTANGLE` adentro de '`|...|`'.)

Los siguientes códigos de control introducen "textos de control," ("*control texts,*") cuales terminan con el siguiente '`@>`'. El '`@>`' de cierre deberá estar en la misma línea del archivo `CWEB` como la línea donde el texto de control comenzó. El contexto de cada uno de estos códigos de control al `@>` correlativo es restringido.

`@^` $[TMCc]$ $*$ El texto de control que sigue, al siguiente '`@>`', entrará en el índice junto con los identificadores del programa C; este texto aparecerá en tipo de letra romana (*roman type*). Por ejemplo, para poner la frase "dependencias del sistema" dentro del índice que es la salida hecha por `CWEAVE`, escribe '`@^dependencias del sistema@>`' en cada sección que tu desees indexar como dependiente del sistema.

`@.` $[TMCc]$ $*$ El texto de control que sigue será ingresado al índice en `tipo de letra de maquina de escribir` (*typewriter type*).

`@:` $[TMCc]$ $*$ El texto de control que sigue será ingresado al índice en un formato controlado por la macro TEX '`\9`', el cual lo definirás a gusto.

`@t` [MCc] ∗ El texto de control que sigue será puesto en un TEX `\hbox` y formateado junto con el programa C vecino. Este texto es ignorado por `CTANGLE`, pero puede ser usado para varios propósitos con `CWEAVE`. Por ejemplo, puedes hacer comentarios que mezclen C y matemáticas clásicas, como en '$size < 2^{15}$', al escribir '`|size < 2@t$^{15}$@>|`'.

`@=` [MCc] ∗ El texto de control que sigue será pasado textualmente al programa C.

`@q` [$LTMCc$] ∗ El texto de control que sigue será totalmente ignorado, solamente es un comentario para lectores del archivo `CWEB`. Un archivo que es intentado ser incluido en el limbo, con `@i`, se puede identificar con comentarios `@q`. Otro uso es balancear paréntesis desbalanceados en cadenas de caracteres de C, para que de esta forma no se caiga abruptamente el agrupador de paréntesis de tu editor.

`@!` [$TMCc$] ∗ El número de sección en una entrada de índice sera subrayada si '`@!`' precede inmediatamente el identificador o el texto de control que está siendo indexado. Esta convención es usada para distinguir las secciones donde un identificador es definido o donde es explicado de una forma especial, desde las secciones donde fue usado. Una palabra reservada o un identificador de largo uno no será indexado, excepto por las entradas subrayadas. Un '`@!`' es implícitamente insertado por `CWEAVE` cuando un identificador está siendo definido o declarado en código C; por ejemplo, la definición

$$\textbf{int}\ array[max\_dim],\ count = old\_count;$$

hace que los nombres *array* y *count* tengan una entrada subrayada en el índice. Etiquetas de sentencias, definiciones de funciones como $main(\textbf{int}\ argc, \textbf{char}\ *argv[\,])$ y definiciones **typedef** también implican subrayado. Una definición de función de viejo-estilo (sin prototipo) no define sus argumentos; estos, sin embargo, serán considerados para ser definidos (i.e., sus entradas de índices serán subrayadas) si sus tipos son declarados antes del cuerpo de la función de la forma usual (p. ej., '**int** *argc*; **char** $*argv[\,]$; $\{\ldots\}$'). Así `@!` no es necesitado muy a menudo, excepto en construcciones no usuales o en casos como

```
enum boolean {@!false, @!true};
```

aquí `@!` entrega los mejores resultados porque constantes individuales enumeradas por `enum` no son subrayadas automáticamente en el índice en sus puntos de definición.

Ahora nos cambiamos a códigos de control que afectan sólo la operación de `CTANGLE`.

`@'` [MCc] Este código de control es peligroso porque tiene significados bastante diferentes en `CWEB` y en el original `WEB`. En `CWEB` produce la constante decimal correspondiente al código ASCII para una cadena de caracteres de largo 1 (p. ej., `@'a'` es `97` y `@'\t'` es `9` al usarlo con `CTANGLE`). Podrías querer usarlo si necesitas trabajar con ASCII en una máquina no ASCII; pero en la mayoría de los casos las convenciones C de `<ctype.h>` son adecuadas para programación independiente del conjunto de caracteres (*character-set-independent*).

`@&` [MCc] La operación `@&` causa que lo que esté a su izquierda sea adyacente a lo que está a su derecha, en la salida de C. Ningún espacio o salto de línea separará estos dos ítemes.

`@l` [L] Programadores `CWEB` tienen la opción de usar código de caracteres de 8-bit del a-menudo-prohibido rango de 128 a 255 en el texto TEX; estos son también permitidos en cadenas de caracteres e incluso en identificadores del programa C. En varias extensiones del estándar ASCII básico, los códigos más alto de 8-bit corresponden a las letras acentuadas, letras de alfabetos no latinos, etc. Cuando esos caracteres aparecen en identificadores, `CTANGLE` deberá reemplazarlos por caracteres alfanuméricos estándares ASCII o `_`, a fin de generar código C legal. Hace esto por medio de una tabla de transcripción, la cual por omisión asocia la cadena de caracteres `Xab` al carácter con el código ASCII $^{\#}ab$ (donde a y b son dígitos hexadecimales y $a \geq 8$). Al colocar la construcción `@l␣ab␣newstring` en el limbo, le estás diciendo a `CTANGLE` que reemplace este carácter por `newstring`. Por ejemplo, El código ISO Latin-1 para la letra 'ü' es $^{\#}$`FC` (o `'\374'`), y `CTANGLE` normalmente cambiará este código por la secuencia de tres-caracteres `XFC` si aparece en un identificador. Si dices `@l fc ue`, el código será transcrito a `ue`.

`CWEAVE` pasa caracteres de 8-bit directo a TEX sin transcripción; por lo tanto, TEX deberá prepararse para recibirlos. Si estás formateando todos sus identificadores no estándares como secuencias de control "personalizadas" ("*custom*"), deberás hacer que TEX trate todos los caracteres como letras. Si no deberás o hacer tus códigos de 8-bit "activos" en TEX, o cargas las fuentes que contienen los caracteres especiales que necesitas en las posiciones correctas. (La fuente seleccionada por la secuencia de control TEX `\it` es usada para identificadores.) Busca paquetes (*packages*) de macros especiales diseñadas para usuarios `CWEB` en tu lenguaje; o si eres valiente escribe uno.

Los siguientes ocho códigos de control (concretamente '`@,`', '`@/`', '`@|`', '`@#`', '`@+`', '`@;`', '`@[`' y '`@]`') no tienen efecto en la salida del programa C hecha por `CTANGLE`; ellos simplemente ayudan a improvisar la legibilidad del C con TEX-formateado que es la salida hecha por `CWEAVE`, en inusuales circunstancias. El método de formateo incrustado de `CWEAVE` es realmente bueno cuando transa con texto C sintácticamente correcto, pero es incapaz de manejar todos los casos posibles, porque deberá transar con fragmentos de texto relacionados con macros y nombres de sección; estos fragmentos no necesariamente obedecen a la sintaxis de C. Aunque `CWEB` te permita sobre-escribir el formateo automático, tu mejor estrategia es no preocuparse sobre esas cosas hasta que hayas visto lo que `CWEAVE` produce automáticamente, ya que probablemente necesitarás hacer solamente unas pocas correcciones cuando estás retocando tu documentación.

`@,` [*MCc*] Este código de control inserta un espacio delgado en la salida de `CWEAVE`. Algunas veces necesitas este espacio extra si estás usando macros de una forma inusual, p. ej., si dos identificadores son adyacentes.

`@/` [*MC*] Este código de control causa un salto de línea que ocurrirá dentro del programa formateado C hecho por `CWEAVE`. Saltos de líneas son elegidos automáticamente por TEX según un esquema que funciona 99% pero algunas veces preferirá forzar un salto de línea así que el programa es segmentado según un criterio lógico más que uno visual. Si un comentario sigue, di '`@/@,`' para saltar la línea antes del comentario.

`@|` [*MC*] Este código de control especifica una salto de línea opcional en el medio de una expresión. Por ejemplo, si tienes una expresión larga en el lado derecho de una sentencia de asignación, puedes usar '`@|`' para especificar puntos-de-interrupción (*breakpoints*) más lógicos que los que TEX podría escoger en terrenos visuales.

`@#` [*MC*] Este código de control fuerza a un salto de línea, como lo hace `@/`, y también causa un poco de espacio extra que aparece entre las líneas de este corte (*break*). Podrías usarlo, por ejemplo, entre grupos de definiciones de macro que están lógicamente separadas, pero dentro de la misma sección. `CWEB` automáticamente inserta este espacio extra entre funciones, entre declaraciones externas y funciones y entre declaraciones y sentencias dentro de una función.

`@+` [*MC*] Este código de control cancela un salto de línea que podría de otra manera ser insertado por `CWEAVE`, p. ej., antes de la palabra '**else**', si quieres poner una construcción pequeña if–else en una sola línea. Si dices '`{@+`' al principio de una sentencia compuesta que es el cuerpo de una función, la primera declaración o sentencia de la función aparecerá en la misma línea como la llave izquierda y se le hará sangría por la misma cantidad que la segunda declaración o sentencia de la línea siguiente.

`@;` [*MC*] Este código de control es tratado como un punto y coma, para propósitos de formateo, excepto que es invisible. Puedes usarlo, por ejemplo, después de un nombre de sección o macro cuando el texto C representado por esa sección o macro es una sentencia compuesta o termina en un punto y coma. Considere construcciones como

```
if (condition) macro @;
else break;
```

donde *macro* es definida como una sentencia compuesta (encerrada entre llaves). Esta es una infelicidad muy famosa de la sintaxis de C.

`@[` [*MC*] Ve `@]`.

`@]` [*MC*] Pon paréntesis `@[...@]` alrededor de texto de un programa texto que supuestamente `CWEAVE` va a formatear como una expresión, si ya no lo ha hecho. (Esto ocasionalmente aplica a argumentos macro inusuales.) También inserta '`@[@]`' entre un nombre de tipo simple y un paréntesis izquierdo cuando se declara un puntero a una función, como en

```
int @[@] (*f)();
```

si no `CWEAVE` confundirá la primera parte de la declaración con la expresión C++ '**int**$(*f)$'. Otro ejemplo, para la gente que quiere usar los comandos de bajo nivel `#define` entre medio del código C y la definición empieza con una conversión (*cast*):

```
#define foo @[(int)(bar)@]
```

Los códigos de control restantes determinan la entrada que `CWEB` ve.

`@x @y @z` [*change_file*] `CWEAVE` y `CTANGLE` están diseñados para trabajar con dos archivos de entrada, llamados *web_file* y *change_file*, donde *change_file* contiene datos que sobre-escriben porciones seleccionadas del *web_file*. El texto unido resultante es realmente a lo que se le ha denominado archivo `CWEB` en las otras partes de este reporte.

Así es como funciona: el archivo con cambios consiste de cero o más "cambios," donde un cambio tiene la forma '`@x`⟨old lines⟩`@y`⟨new lines⟩`@z`'. Los códigos de control especiales `@x`, `@y`, `@z`, los cuales sólo están permitidos en archivos de cambio, deberán aparecer al principio de una línea; el resto de tal línea es ignorado. ⟨old lines⟩ representa material que exactamente hace una correlación entre líneas consecutivas del *web_file*; ⟨new lines⟩ representa cero o más líneas que están supuestamente para reemplazar las antiguas. Siempre que la primera "línea antigua" ("*old line*") de un cambio es encontrada para hacer una correlación con una línea en el *web_file*, todas las otras líneas de ese cambio deberán hacer una correlación también.

Entre cambios, antes del primer cambio y después del último, el archivo de ellos puede tener cualquier número de líneas que no comiencen con '`@x`', '`@y`', o '`@z`'. Tales líneas son evitadas y no usadas para propósitos de correlación.

Esta característica de entrada doble (*dual-input*) es útil cuando se trabaja con un archivo maestro `CWEB` que ha sido recibido desde otro lado (p. ej., `tangle.w` o `weave.w` o `tex.web`), cuando los cambios son atractivos de personalizar el programa para tu sistema de computadora. Serás capaz de depurar tus cambios de sistema-dependiente sin golpear el archivo *web* maestro; y cuando tus cambios estén funcionando, serás capaz de incorporarlos sin inconvenientes dentro de nuevos entregables del archivo *web* maestro que podrías haber recibido de vez en cuando.

`@i` [*web_file*] Además *web_file* en si mismo puede ser una combinación de varios archivos. Cuando algún `CWEAVE` o `CTANGLE` está leyendo un archivo y encuentra el código de control `@i` al principio de la línea, interrumpe la lectura normal y empieza a mirar en el archivo llamado después de `@i`, tanto como el preprocesador C haga cuando encuentre una línea `#include`. Después que el archivo incluido haya sido leído completamente, el programa se regresa a la siguiente línea del archivo original. El nombre de archivo que sigue a `@i` puede ser rodeado por caracteres `"`, pero tales delimitadores son opcionales. Archivos incluidos pueden anidar (*nest*).

Archivos de cambio tienen líneas que comienzan con `@i`. De esta forma puedes reemplazar un archivo incluido con otro. Conceptualmente, el mecanismo de reemplazo descrito arriba hace su trabajo primero y su salida es chequeada por líneas `@i`. Si `@i foo` ocurre entre `@y` y `@z` en un archivo de cambio, líneas individuales del archivo `foo` los archiva y los archivos incluidos no son cambiables; pero cambios pueden ser hechos a líneas desde archivos que fueron incluidos por una entrada sin cambios.

En sistemas UNIX (y otros que soporten variables de entorno), si la variable de entorno `CWEBINPUTS` es configurada o si la bandera del compilador (*compiler flag*) del mismo nombre fue definido a tiempo de compilación, `CWEB` buscará archivos incluidos en el directorio llamado así, si no puede encontrarlos en el directorio actual.

Características adicionales y advertencias

1. En ciertas instalaciones de `CWEB` que tienen un conjunto de caracteres extendido, los caracteres '`↑`', '`↓`', '`→`', '`≠`', '`≤`', '`≥`', '`≡`', '`∨`', '`∧`', '`⊂`' y '`⊃`' pueden ser escritos como abreviaciones para '`++`', '`--`', '`->`', '`!=`', '`<=`', '`>=`', '`==`', '`||`', '`&&`', '`<<`' y '`>>`', respectivamente.

2. Si tienes un conjunto de caracteres extendido, puedes usarlo sólo con restricciones mínimas, como fue discutido arriba con las reglas de `@l`. pero deberás adherir los caracteres estándares ASCII si quieres escribir programas que serán útiles para todas las pobres almas allá afuera quienes no tienen conjuntos de caracteres extendidos.

3. El archivo de salida TEX hecho por `CWEAVE` es dividido en líneas de un máximo de 80 caracteres cada uno. Cuando el texto TEX está siendo copiado, los saltos de líneas existentes son copiados también. Si no estás haciendo algo muy peliagudo, `CWEAVE` reconocerá cuando un comentario TEX está siendo divido entre dos o más líneas, y agregará '`%`' al principio de tales comentarios ininterrumpidos.

4. Texto C es traducido por un procedimiento "de abajo arriba" ("*bottom up*") que identifica cada componente léxico (*token*) como una "parte del discurso" ("*part of speech*") y combina las partes del discurso en frases más y más grandes tanto como sea posible según una gramática especial que es explicada en la documentación de `CWEAVE`. Es fácil de aprender el esquema de traducción para construcciones simples como identificadores únicos y expresiones cortas, al mirar unos pocos ejemplos de lo que hace `CWEAVE`, pero el

mecanismo general es algo complejo porque deberá encargarse mucho más que del mismo C. Además la salida contiene código incrustado que causa al TEX hacer sangrías y saltos de líneas cuando sea necesario, dependiendo de las fuentes usadas y los anchos de páginas deseados. Para mejores resultados es sensato evitar encerrar textos largos C en `|...|`, desde que los códigos de sangría y saltos de líneas son omitidos cuando el texto `|...|` es traducido desde C a TEX. Apégate a expresiones o sentencias simples. Si un comando de preprocesador C está dentro de `|...|`, el `#` que lo introduce deberá estar al principio de una línea, o `CWEAVE` no lo imprimirá correctamente.

5. Comentarios son permitidos en texto `|...|`. Después de un '`|`' señala el cambio desde texto TEX al texto C, el siguiente '`|`' que no es parte de una cadena de caracteres o texto de control o nombre de sección termina el texto C.

6. Un comentario deberá tener correctamente ocurrencias anidadas de llaves de la izquierda y derecha, de otro modo `CWEAVE` se quejará. Pero intenta balancear las llaves, de tal manera que TEX no fallará en demasía.

7. Cuando estás depurando un programa y decides omitir algún código C, simplemente NO "le saques el comentario." ("*comment it out.*") Esos comentarios no están en el espíritu de la documentación de `CWEB`; aparecerán a los lectores como si fuesen explicaciones de instrucciones no comentadas. Además, comentarios de un programa deberán ser texto válido TEX; de ahí `CWEAVE` se confundirá si encierras sentencias C en `/*...*/` en vez de estar en `/*|...|*/`. Si debes sacar los comentarios de código C, puedes rodearlos con comandos de preprocesador como `#if 0==1` y `#endif`.

8. La característica `@f` te permite definir un identificador que actúe como otro, y estas definiciones de formato son llevadas a cabo secuencialmente. En general, un identificador dado tiene sólo un formato impreso a través del documento entero y este formato es usado incluso antes del `@f` que lo define. La razón es que `CWEAVE` opera en dos pasos; procesa los `@f` y hace referencias cruzadas en el primer paso y genera la salida en el segundo. (Sin embargo, identificadores que implícitamente tienen un formato en negrita, gracias a la declaración `typedef`, no hagas caso de esta regla; ellos son impresas diferentemente antes y después del relevante `typedef`. Esto es inoportuno y difícil de arreglar. Puedes evitar eludir la restricción al decir algo como, '`@s foo int`', antes o después del `typedef`.)

9. Algunas veces es deseable insertar espacios en blanco entre el código C formateado que es más general que el espacio delgado proveído por '`@,`'. La característica `@t` puede ser usada para este propósito; p. ej., '`@t\hskip 1in@>`' dejará una pulgada de espacio en blanco. Es más, '`@t\4@>`' puede ser usado para retroceder un espacio en una unidad de sangría, ya que la secuencia de control `\4` es definida en `cwebmac` para ser ese espacio de retroceso. (Esta secuencia de control es usada, por ejemplo, al principio de líneas que contengan sentencias etiquetadas, de esta forma la etiqueta quedará puesta un poco a la izquierda.)

También puedes usar '`@t}\3{-5@>`' para forzar un corte al medio de una expresión.

10. Cada identificador en `CWEB` tiene una convención de formateo única. Por lo tanto, no deberías usar el mismo identificador para denotar, digamos a un nombre de tipo (*type name*) y parte de `struct`, aunque C permita esto.

Corriendo los programas

La línea de comando UNIX para `CTANGLE` es

```
ctangle [options] web_file[.w] [{change_file[.ch]|-} [out_file]]
```

y las mismas convenciones se aplican a `CWEAVE`. Si aparece '`-`' o no hay archivo de cambio especificado, entonces el archivo de cambio es nulo. Las extensiones `.w` y `.ch` son agregadas solamente si los nombres de archivos dados no contienen puntos. Si el archivo *web* definido de esta forma no puede ser encontrado, la extensión `.web` será intentada. Por ejemplo, '`cweave cob`' intentará leer `cob.w`; si es que falla, intentará `cob.web` antes de rendirse. Si no se ha especificado nombre de archivo, el nombre del archivo de salida C hecho por `CTANGLE` es obtenido al agregarle la extensión `.c`; el nombre del archivo de salida TEX hecho por `CWEAVE` tendrá la extensión `.tex`. Archivos índices hechos por `CWEAVE` reemplazan `.tex` por `.idx` y `.scn`.

Los programadores que les gusta el estilo lacónico podrían elegir para configurar sus consolas de tal manera que '`wv`' se expanda a '`cweave -bhp`'; esto suprimirá la mayoría de la salida terminal desde `CWEAVE` excepto por los mensajes de error.

Las opciones son introducidas por el signo `-`, para desactivar una opción (*off*), o por un signo `+` para activar (*on*). Por ejemplo, '`-fb`' desactiva a las opciones `f` y `b`; '`+s`' activa a la opción `s`. Las opciones

pueden ser especificadas antes ó después de los nombres de archivos, o ambos. Las opciones siguientes son implementadas actualmente:

`b` imprime una línea tipo pancarta ("*banner*") al principio de la ejecución. (activa por omisión.)
`e` Encierra material C formateado por `CWEAVE` dentro de llaves `\PB{...}`, así esos enganches especiales pueden ser usados. (Desactiva por omisión; no tiene efecto en `CTANGLE`.)
`f` Fuerza saltos de líneas después de cada sentencia C formateada por `CWEAVE`. (Activa por omisión; `-f` ahorra papel pero luce menos al estilo C para ciertas personas.) (No tiene efecto en `CTANGLE`.)
`h` Imprime un mensaje feliz al concluir exitosamente un programa. (Activa por omisión.)
`p` Da reportes de progreso por mientras el programa corre. (Activo por omisión.)
`s` Muestra estadísticas sobre el uso de memoria después que el programa termine. (desactivo por omisión.) Si tienes archivos `CWEB` largos o secciones, podrías necesitar ver cuan cerca te acercas para exceder la capacidad de `CTANGLE` y/o `CWEAVE`.
`x` Incluye índices y una tabla de contenidos en el archivo de salida TEX hecho por `CWEAVE`. (Activo por omisión.) (No tiene efecto en `CTANGLE`.)

Detalles más a fondo sobre formateo

Podría no gustarte la forma que `CWEAVE` se encarga de ciertas situaciones. Si estás desesperado, puedes personalizar `CWEAVE` al cambiar su gramática. Esto significa cambiar el código fuente, una tarea que podrías encontrarla entretenida. Una tabla de reglas gramaticales aparece en el listado del código de `CWEAVE` y puedes hacer una copia separada de esa tabla al copiar el archivo `prod.w` que se encuentra en las fuentes de `CWEB` y ejecutando '`cweave -x prod`', seguido por '`tex prod`'.

Puedes ver exactamente como `CWEAVE` analiza sintácticamente (*parsing*) tu código C al precederle con la línea '`@ @c @2`'. (El código de control '`@2`' se convierte en modo de "fisgón" ("*peeping*") y '`@0`' lo desactiva.) Por ejemplo, si tu corres `CWEAVE` con el archivo

```
@ @c @2
main (argc,argv)
char **argv;
{ for (;argc>0;argc--) printf("%s\n",argv[argc-1]); }
```

obtienes las siguientes incoherencias en tu pantalla:

```
[...]
4:*exp ( +exp+ )...
11:*exp +exp+ int...
5:*+exp+ int +unorbinop+...
[...]
60: +fn_decl+*+{+ -stmt- +}-
55:*+fn_decl+ -stmt-
52:*+function-
[...]
```

La primera línea dice que la regla gramatical 4 recién ha sido aplicada y `CWEAVE` actualmente tiene en su memoria una secuencia de pedazos de código TEX llamados "sobras" ("*scraps*") que son respectivamente del tipo *exp* (para expresión), paréntesis abiertos, *exp* de nuevo, paréntesis cerrados y otras sobras que aun no han sido considerados por el analizador sintáctico (*parser*). (Los signos `+` y `-` estipulan que TEX debería está dentro o fuera del modo matemático en los límites de la sobra. El `*` muestra la posición actual del analizador.) Luego la regla 11 es aplicada y la secuencia (*exp*) se convierte en *exp* y así sucesivamente. Al final el texto C se ha convertido en una gran sobra del tipo de *function*.

Algunas veces no todo funciona tan bien y obtienes un montón de líneas abultadas. Esto quiere decir que `CWEAVE` no podrá digerir alguna cosa de tu código C. Por ejemplo, supón '`@<Argument definitions@>`' ha aparecido en vez de '`char **argv;`' en el programa de arriba. Luego `CWEAVE` se habría desconcertado de alguna manera, ya que piensa que los nombres de sección son sólo *exp*s. Así le diría al TEX formatear '⟨ Argument declarations 2 ⟩' en la misma línea tal como '*main*(*argc*, *argv*)'. En este caso deberás ayudar al `CWEAVE` poniendo '`@/`' después de '`main(argc,argv)`'.

CWEAVE automáticamente inserta un poco de espacio extra entre las declaraciones y la primera aparente sentencia de un bloque. Una forma de derrotar este espacio localmente es

```
int x;@+@t}\6{@>
@<Other locals@>@;@#
```

el '@#' pondrá espacio extra después de '⟨ Other locals ⟩'.

Hipertexto e hiperdocumentación

Por supuesto mucha gente notó analogías entre CWEB y la World Wide Web. Las macros CWEB son realmente configuradas de tal manera que la salida de CWEAVE pueda ser convertida fácilmente a Portable Document Format, con hipervínculos pinchables que pueden ser leídos por el Acrobat Reader de Adobe, al usar un programa open source de mucha divulgación llamado dvipdfm desarrollado por Mark A. Wicks. Después usando CWEAVE para convertir cob.w en cob.tex, puedes preparar y ver una versión de hipertexto del programa al dar los comandos

```
tex "\let\pdf+ \input cob"
dvipdfm cob
acroread cob.pdf
```

En vez de invocar TEX de la forma tradicional. (Gracias a Hans Hagen, César Augusto Rorato Crusius y Julian Gilbey por las macros que hacen este trabajo.) por otro lado (gracias a Hàn Thế Thành y Andreas Scherer) puedes generar cob.pdf en un paso simplemente al decir 'pdftex cob'.

Un sistema más elaborado llamado CTWILL, el cual extiende las típicas referencias cruzadas de CWEAVE al preparar vínculos desde los usos de identificadores a sus definiciones, está también disponible, siempre y cuando que estés deseando trabajar un poco más duro en los casos donde un identificador es definido exponencialmente. CTWILL se intentó primeramente hacer como salida para un libro impreso, pero sus principios también pueden ser usados para hipertexto. Véase Capítulo 11 de *Digital Typography* por D. E. Knuth (1999) y los programas fuentes en ftp://ftp.cs.stanford.edu/pub/ctwill.

Apéndices

Como un ejemplo de un programa real escrito en CWEB, Apéndice A contiene un pasaje del programa CWEB. El lector que examine los listados cuidadosamente en este apéndice observará las ideas básicas de CWEB.

Apéndice B es el archivo que configura TEX para aceptar la salida de CWEAVE y Appendix C discute como usar algunas de estas macros para variar los formatos de salida.

Una versión "larga" de este manual, puede ser producida desde las fuentes de CWEB vía el comando UNIX make fullmanual, también contiene apéndices D, E y F, los cuales exhiben el código fuente de CTANGLE y CWEAVE.

Apéndice A: Pasajes de un Programa CWEB

Este apéndice se constituye de cuatro listados. El primero muestra la entrada de CWEB que generó las secciones 12 hasta la 15 del archivo common.w, las cuales contienen rutinas comunes a CWEAVE y CTANGLE. Nóteseque algunos de las líneas están intentando mostrar la estructura del programa; la sangría es ignorada por CWEAVE y CTANGLE, pero los usuarios encuentran que los archivos CWEB son bastante legibles si es que tienen tales sangrías.

El segundo y tercer listado muestra partes correspondientes de la salida de código C hecha por CTANGLE y de la correspondiente salida de código TEX hecha por CWEAVE, cuando corres con common.w. El cuarto listado muestra como la salida luce cuando es impresa.

```
@ Procedure |prime_the_change_buffer|
sets |change_buffer| in preparation for the next matching operation.
Since blank lines in the change file are not used for matching,
 we have
|(change_limit==change_buffer

 && !changing)| if and only if
the change file is exhausted.
 This procedure is called only when
|changing| is 1; hence error messages will be reported correctly.

@c
void
prime_the_change_buffer()
{
  change_limit=change_buffer; /* this value
 is used if the change file ends */
  @<Skip over comment lines in the change file; |return| if end of file@>;
  @<Skip to the next nonblank line; |return| if end of file@>;
  @<Move |buffer| and |limit| to |change_buffer| and |change_limit|@>;
}

@ While looking for a line that begins with \.{@@x} in the change file, we
allow lines that begin with \.{@@}, as long as they don't begin with \.{@@y},
\.{@@z}, or \.{@@i} (which would probably mean that the change file is fouled up).

@<Skip over comment lines in the change file...@>=
while(1) {
  change_line++;
  if (!input_ln(change_file)) return;
  if (limit<buffer+2) continue;
  if (buffer[0]!='@@') continue;
  if (xisupper(buffer[1])) buffer[1]=tolower(buffer[1]);
  if (buffer[1]=='x') break;
  if (buffer[1]=='y' || buffer[1]=='z' || buffer[1]=='i') {
    loc=buffer+2;
    err_print("! Missing @@x in change file");
@.Missing @@x...@>
  }
}

@ Here we are looking at lines following the \.{@@x}.

@<Skip to the next nonblank line...@>=
do {
  change_line++;
  if (!input_ln(change_file)) {
    err_print("! Change file ended after @@x");
@.Change file ended...@>
    return;
  }
} while (limit==buffer);

@ @<Move |buffer| and |limit| to |change_buffer| and |change_limit|@>=
{
  change_limit=change_buffer-buffer+limit;
  strncpy(change_buffer,buffer,limit-buffer+1);
}
```

Acá está la porción de código C generado por CTANGLE que corresponde a la fuente en la página anterior. Nota que las secciones 13, 14 y 15 han sido enredadas dentro de la sección 12.

```
/*:9*//*12:*/
#line 247 "common.w"

void
prime_the_change_buffer()
{
change_limit= change_buffer;
/*13:*/
#line 261 "common.w"

while(1){
change_line++;
if(!input_ln(change_file))return;
if(limit<buffer+2)continue;
if(buffer[0]!='@')continue;
if(xisupper(buffer[1]))buffer[1]= tolower(buffer[1]);
if(buffer[1]=='x')break;
if(buffer[1]=='y'||buffer[1]=='z'||buffer[1]=='i'){
loc= buffer+2;
err_print("! Missing @x in change file");

}
}

/*:13*/
#line 252 "common.w"
;
/*14:*/
#line 278 "common.w"

do{
change_line++;
if(!input_ln(change_file)){
err_print("! Change file ended after @x");

return;
}
}while(limit==buffer);

/*:14*/
#line 253 "common.w"
;
/*15:*/
#line 288 "common.w"

{
change_limit= change_buffer-buffer+limit;
strncpy(change_buffer,buffer,limit-buffer+1);
}

/*:15*/
#line 254 "common.w"
;
}

/*:12*//*16:*/
```

Aquí está el pasaje correspondiente de common.tex.

```
\M{12}Procedure \PB{\\{prime\_the\_change\_buffer}}
sets \PB{\\{change\_buffer}} in preparation for the next matching operation.
Since blank lines in the change file are not used for matching, we have
\PB{$(\\{change\_limit}\E\\{change\_buffer}\W\R\\{changing})$} if and only if
the change file is exhausted. This procedure is called only when
\PB{\\{changing}} is 1; hence error messages will be reported correctly.

\Y\B\&{void} \\{prime\_the\_change\_buffer}(\,)\1\1\2\2\6
${}\{{}$\1\6
${}\\{change\_limit}\K\\{change\_buffer}{}$;\C{ this value is used if the
change file ends }\6
\X13:Skip over comment lines in the change file; \PB{\&{return}} if end of file%
\X;\6
\X14:Skip to the next nonblank line; \PB{\&{return}} if end of file\X;\6
\X15:Move \PB{\\{buffer}} and \PB{\\{limit}} to \PB{\\{change\_buffer}} and %
\PB{\\{change\_limit}}\X;\6
\4${}\}{}$\2\par
\fi

\M{13}While looking for a line that begins with \.{@x} in the change file, we
allow lines that begin with \.{@}, as long as they don't begin with \.{@y},
\.{@z}, or \.{@i} (which would probably mean that the change file is fouled
up).

\Y\B\4\X13:Skip over comment lines in the change file; \PB{\&{return}} if end
of file\X${}\E{}$\6
\&{while} (\T{1})\5
${}\{{}$\1\6
${}\\{change\_line}\PP;{}$\6
\&{if} ${}(\R\\{input\_ln}(\\{change\_file})){}$\1\5
\&{return};\2\6
\&{if} ${}(\\{limit}<\\{buffer}+\T{2}){}$\1\5
\&{continue};\2\6
\&{if} ${}(\\{buffer}[\T{0}]\I\.{'@'}){}$\1\5
\&{continue};\2\6
\&{if} (\\{xisupper}(\\{buffer}[\T{1}]))\1\5
${}\\{buffer}[\T{1}]\K\\{tolower}(\\{buffer}[\T{1}]);{}$\2\6
\&{if} ${}(\\{buffer}[\T{1}]\E\.{'x'}){}$\1\5
\&{break};\2\6
\&{if} ${}(\\{buffer}[\T{1}]\E\.{'y'}\V\\{buffer}[\T{1}]\E\.{'z'}\V\\{buffer}[%
\T{1}]\E\.{'i'}){}$\5
${}\{{}$\1\6
${}\\{loc}\K\\{buffer}+\T{2};{}$\6
\\{err\_print}(\.{"!\ Missing\ @x\ in\ cha}\)\.{nge\ file"});\6
\4${}\}{}$\2\6
\4${}\}{}$\2\par
\U12.\fi

\M{14}Here we are looking at lines following the \.{@x}.

\Y\B\4\X14:Skip to the next nonblank line; \PB{\&{return}} if end of file\X${}%
\E{}$\6
\&{do}\5
${}\{{}$\1\6
${}\\{change\_line}\PP;{}$\6
\&{if} ${}(\R\\{input\_ln}(\\{change\_file})){}$\5
${}\{{}$\1\6
\\{err\_print}(\.{"!\ Change\ file\ ended}\)\.{\ after\ @x"});\6
\&{return};\6
\4${}\}{}$\2\6
\4${}\}{}$\2\5
\&{while} ${}(\\{limit}\E\\{buffer}){}$;\par
\U12.\fi

\M{15}\B\X15:Move \PB{\\{buffer}} and \PB{\\{limit}} to \PB{\\{change\_buffer}}
and \PB{\\{change\_limit}}\X${}\E{}$\6
${}\{{}$\1\6
${}\\{change\_limit}\K\\{change\_buffer}-\\{buffer}+\\{limit};{}$\6
${}\\{strncpy}(\\{change\_buffer},\39\\{buffer},\39\\{limit}-\\{buffer}+%
\T{1});{}$\6
\4${}\}{}$\2\par
\Us12\ET16.\fi
```

Y acá está el mismo pasaje como luce compuesto tipográficamente.

12. Procedure *prime_the_change_buffer* sets *change_buffer* in preparation for the next matching operation. Since blank lines in the change file are not used for matching, we have (*change_limit* ≡ *change_buffer* ∧ ¬*changing*) if and only if the change file is exhausted. This procedure is called only when *changing* is 1; hence error messages will be reported correctly.

```
void prime_the_change_buffer( )
{
  change_limit = change_buffer;     /* this value is used if the change file ends */
  ⟨ Skip over comment lines in the change file; return if end of file 13 ⟩;
  ⟨ Skip to the next nonblank line; return if end of file 14 ⟩;
  ⟨ Move buffer and limit to change_buffer and change_limit 15 ⟩;
}
```

13. While looking for a line that begins with `@x` in the change file, we allow lines that begin with `@`, as long as they don't begin with `@y`, `@z`, or `@i` (which would probably mean that the change file is fouled up).

```
⟨ Skip over comment lines in the change file; return if end of file 13 ⟩ ≡
  while (1) {
    change_line++;
    if (¬input_ln(change_file)) return;
    if (limit < buffer + 2) continue;
    if (buffer[0] ≠ '@') continue;
    if (xisupper(buffer[1])) buffer[1] = tolower(buffer[1]);
    if (buffer[1] ≡ 'x') break;
    if (buffer[1] ≡ 'y' ∨ buffer[1] ≡ 'z' ∨ buffer[1] ≡ 'i') {
      loc = buffer + 2;
      err_print("! Missing @x in change file");
    }
  }
```

This code is used in section 12.

14. Here we are looking at lines following the `@x`.

```
⟨ Skip to the next nonblank line; return if end of file 14 ⟩ ≡
  do {
    change_line++;
    if (¬input_ln(change_file)) {
      err_print("! Change file ended after @x");
      return;
    }
  } while (limit ≡ buffer);
```

This code is used in section 12.

15.

```
⟨ Move buffer and limit to change_buffer and change_limit 15 ⟩ ≡
  {
    change_limit = change_buffer − buffer + limit;
    strncpy(change_buffer, buffer, limit − buffer + 1);
  }
```

This code is used in sections 12 and 16.

Apéndice B: El archivo `cwebmac.tex`

Éste es el archivo que extiende el formato "TEX plano" ("*plain TEX*") para apoyar las características necesitadas por la salida de `CWEAVE`.

```
% standard macros for CWEB listings (in addition to plain.tex)
% Version 3.63 --- January 2001
\ifx\documentstyle\undefined\else\endinput\fi % LaTeX will use other macros
\xdef\fmtversion{\fmtversion+CWEB3.63}
\newif\ifpdf
\ifx\pdf+\pdftrue\fi
% Uncomment the following line if you want PDF goodies to be the default
%\ifx\pdf-\else\pdftrue\fi
\def\pdflinkcolor{0 0 1} % the RGB values for hyperlink color
\newif\ifpdftex
\ifx\pdfoutput\undefined \pdftexfalse
\else \pdftextrue \pdfoutput=1 \pdfcompresslevel=9 \input pdfcolor \fi
\newif\ifacro \ifpdf\acrotrue\fi \ifpdftex\acrotrue\fi

\let\:=\. % preserve a way to get the dot accent
 % (all other accents will still work as usual)

\parskip 0pt % no stretch between paragraphs
\parindent 1em % for paragraphs and for the first line of C text

\font\ninerm=cmr9
\let\mc=\ninerm % medium caps
\def\CEE/{{\mc C\spacefactor1000}}
\def\UNIX/{{\mc U\kern-.05emNIX\spacefactor1000}}
\def\TEX/{\TeX}
\def\CPLUSPLUS/{{\mc C\PP\spacefactor1000}}
\def\Cee{\CEE/} % for backward compatibility
\def\9#1{}
 % with this definition of \9 you can say @:sort key}{TeX code@>
 % to alphabetize an index entry by the sort key but format with the TeX code
\font\eightrm=cmr8
\let\sc=\eightrm % for smallish caps (NOT a caps-and-small-caps font)
\let\mainfont=\tenrm
\let\cmntfont\tenrm
%\font\tenss=cmss10 \let\cmntfont\tenss % alternative comment font
\font\titlefont=cmr7 scaled\magstep4 % title on the contents page
\font\ttitlefont=cmtt10 scaled\magstep2 % typewriter type in title
\font\tentex=cmtex10 % TeX extended character set (used in strings)
\fontdimen7\tentex=0pt % no double space after sentences

\def\\#1{\leavevmode\hbox{\it#1\/\kern.05em}} % italic type for identifiers
\def\|#1{\leavevmode\hbox{$#1$}} % one-letter identifiers look better this way
\def\&#1{\leavevmode\hbox{\bf
  \def\_{\kern.04em\vbox{\hrule width.3em height .6pt}\kern.08em}%
  #1\/\kern.05em}} % boldface type for reserved words
\def\.#1{\leavevmode\hbox{\tentex % typewriter type for strings
  \let\\=\BS % backslash in a string
  \let\{=\LB % left brace in a string
  \let\}=\RB % right brace in a string
  \let\~=\TL % tilde in a string
  \let\ =\SP % space in a string
  \let\_=\UL % underline in a string
  \let\&=\AM % ampersand in a string
  \let\^=\CF % circumflex in a string
  #1\kern.05em}}
\def\){{\tentex\kern-.05em}\discretionary{\hbox{\tentex\BS}}{}{}}
\def\AT{@} % at sign for control text (not needed in versions >= 2.9)
\def\ATL{\par\noindent\bgroup\catcode`\_=12 \postATL} % print @l in limbo
\def\postATL#1 #2 {\bf letter \\{\uppercase{\char"#1}}
   tangles as \tentex "#2"\egroup\par}
\def\noATL#1 #2 {}
\def\noatl{\let\ATL=\noATL} % suppress output from @l
\def\ATH{{\acrofalse\X\kern-.5em:Preprocessor definitions\X}}
\let\PB=\relax % hook for program brackets |...| in TeX part or section name
```

```
\chardef\AM=`\& % ampersand character in a string
\chardef\BS=`\\ % backslash in a string
\chardef\LB=`\{ % left brace in a string
\chardef\RB=`\} % right brace in a string
\def\SP{{\tt\char`\ }} % (visible) space in a string
\chardef\TL=`\~ % tilde in a string
\chardef\UL=`\_ % underline character in a string
\chardef\CF=`\^ % circumflex character in a string

\newbox\PPbox % symbol for ++
\setbox\PPbox=\hbox{\kern.5pt\raise1pt\hbox{\sevenrm+\kern-1pt+}\kern.5pt}
\def\PP{\copy\PPbox}
\newbox\MMbox \setbox\MMbox=\hbox{\kern.5pt\raise1pt\hbox{\sevensy\char0
 \kern-1pt\char0}\kern.5pt}
\def\MM{\copy\MMbox}
\newbox\MGbox % symbol for ->
\setbox\MGbox=\hbox{\kern-2pt\lower3pt\hbox{\teni\char'176}\kern1pt}
\def\MG{\copy\MGbox}
\def\MRL#1{\mathrel{\let\K==#1}}
%\def\MRL#1{\KK#1}\def\KK#1#2{\buildrel\;#1\over{#2}}
\let\GG=\gg
\let\LL=\ll
\let\NULL=\Lambda
\mathchardef\AND="2026 % bitwise and; also \& (unary operator)
\let\OR=\mid % bitwise or
\let\XOR=\oplus % bitwise exclusive or
\def\CM{{\sim}} % bitwise complement
\newbox\MODbox \setbox\MODbox=\hbox{\eightrm\%}
\def\MOD{\mathbin{\copy\MODbox}}
\def\DC{\kern.1em{::}\kern.1em} % symbol for ::
\def\PA{\mathbin{.*}} % symbol for .*
\def\MGA{\mathbin{\MG*}} % symbol for ->*
\def\this{\&{this}}

\newbox\bak \setbox\bak=\hbox to -1em{} % backspace one em
\newbox\bakk\setbox\bakk=\hbox to -2em{} % backspace two ems

\newcount\ind % current indentation in ems
\def\1{\global\advance\ind by1\hangindent\ind em} % indent one more notch
\def\2{\global\advance\ind by-1} % indent one less notch
\def\3#1{\hfil\penalty#10\hfilneg} % optional break within a statement
\def\4{\copy\bak} % backspace one notch
\def\5{\hfil\penalty-1\hfilneg\kern2.5em\copy\bakk\ignorespaces}% optional break
\def\6{\ifmmode\else\par % forced break
  \hangindent\ind em\noindent\kern\ind em\copy\bakk\ignorespaces\fi}
\def\7{\Y\6} % forced break and a little extra space
\def\8{\hskip-\ind em\hskip 2em} % no indentation

\newcount\gdepth % depth of current major group, plus one
\newcount\secpagedepth
\secpagedepth=3 % page breaks will occur for depths -1, 0, and 1
\newtoks\gtitle % title of current major group
\newskip\intersecskip \intersecskip=12pt minus 3pt % space between sections
\let\yskip=\smallskip
\def\?{\mathrel?}
\def\note#1#2.{\Y\noindent{\hangindent2em%
    \baselineskip10pt\eightrm#1~\ifacro{\pdfnote#2.}\else#2\fi.\par}}
% The following are pdf macros
\newtoks\toksA \newtoks\toksB \newtoks\toksC \newtoks\toksD
\newcount\countA \countA=0 \newcount\countB \countB=0
\def\thewidth{\the\wd0\space}
\def\theheight{\the\ht0\space}
\def\thedepth{\the\dp0\space}
\ifpdftex
  \ifx\pdfannotlink\undefined\let\pdfannotlink\pdfstartlink\fi% for pdfTeX 0.14
  \def\pdflink#1#2{\hbox{\pdfannotlink attr{/Border [0 0 0]} goto num #1
    \BlueGreen #1\Black\pdfendlink}}
\else\def\pdflink#1#2{\setbox0=\hbox{\special{pdf: bc [ \pdflinkcolor ]}{#1}%
    \special{pdf: ec}}\special{pdf: ann width \thewidth\space height \theheight
      \space depth \thedepth\space << /Type /Annot /Subtype /Link
      /Border [0 0 0] /A << /S /GoTo /D (#2) >> >>}\box0\relax}\fi
```

```
\def\pdfnote#1.{\setbox0=\hbox{\toksA={#1.}\toksB={}\maketoks}\the\toksA}
\def\firstsecno#1.{\setbox0=\hbox{\toksA={#1.}\toksB={}
    \def\makenote{\addtokens{\toksB}{\the\toksC}\def\makenote{\toksD={}
      \toksC={}\let\space\empty}\makenote}\maketoks}}
\def\addtokens#1#2{\edef\addtoks{\noexpand#1={\the#1#2}}\addtoks}
\def\adn#1{\addtokens{\toksC}{#1}\global\countA=1\let\next=\maketoks}
\def\poptoks#1#2|ENDTOKS|{\let\first=#1\toksD={#1}\toksA={#2}}
\def\maketoks{%
    \expandafter\poptoks\the\toksA|ENDTOKS|
    \ifx\first0\adn0
    \else\ifx\first1\adn1 \else\ifx\first2\adn2 \else\ifx\first3\adn3
    \else\ifx\first4\adn4 \else\ifx\first5\adn5 \else\ifx\first6\adn6
    \else\ifx\first7\adn7 \else\ifx\first8\adn8 \else\ifx\first9\adn9
    \else
        \ifnum0=\countA\else\makenote\fi
        \ifx\first.\let\next=\done\else
            \let\next=\maketoks
            \addtokens{\toksB}{\the\toksD}
            \ifx\first,\addtokens{\toksB}{\space}\fi
        \fi
    \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
    \next
}
\def\makenote{\addtokens{\toksB}%
    {\noexpand\pdflink{\the\toksC}{\romannumeral\the\toksC}}\toksC={}\global\countA=0}
\def\done{\edef\st{\global\noexpand\toksA={\the\toksB}}\st}
\def\pdfURL#1#2{\ifpdftex\pdfannotlink attr {/Border [0 0 0]} user {
    /Type /Action /Subtype /Link /A << /S /URI /URI (#2) >>
    }\BlueGreen #1\Black \pdfendlink
  \else \ifpdf{\setbox0=\hbox{\special{pdf: bc [ \pdflinkcolor ]}{#1}%
    \special{pdf: ec}}\special{pdf: ann width \thewidth\space height \theheight
      \space depth \thedepth\space << /Border [0 0 0]
      /Type /Action /Subtype /Link /A << /S /URI /URI (#2) >> >>}\box0\relax}%
  \else #1 ({\tt#2})\fi\fi}
{\catcode`\~=12 \gdef\TILDE/{~}} % ~ in a URL
{\catcode`\_=12 \gdef\UNDER/{_}} % _ in a URL
% End of pdf macros
\def\lapstar{\rlap{*}}
\def\stsec{\rightskip=0pt % get out of C mode (cf. \B)
  \sfcode`;=1500 \pretolerance 200 \hyphenpenalty 50 \exhyphenpenalty 50
  \noindent{\let\*=\lapstar\bf\secstar.\quad}%
  \ifpdftex\smash{\raise\baselineskip\hbox to0pt{%
     \let\*=\empty\pdfdest num \secstar fith}}
  \else\ifpdf\smash{\raise\baselineskip\hbox to0pt{%
     \let\*=\empty\special{%
       pdf: dest (\romannumeral\secstar) [ @thispage /FitH @ypos ]}}}\fi\fi}
\let\startsection=\stsec
\def\defin#1{\global\advance\ind by 2 \1\&{#1 } } % begin `define' or `format'
\def\A{\note{See also section}} % xref for doubly defined section name
\def\As{\note{See also sections}} % xref for multiply defined section name
\def\B{\rightskip=0pt plus 100pt minus 10pt % go into C mode
  \sfcode`;=3000
  \pretolerance 10000
  \hyphenpenalty 1000 % so strings can be broken (discretionary \ is inserted)
  \exhyphenpenalty 10000
  \global\ind=2 \1\ \unskip}
\def\C#1{\5\5\quad$/\ast\,${\cmntfont #1}$\,\ast/$}
\let\SHC\C % "// short comments" treated like "/* ordinary comments */"
%\def\C#1{\5\5\quad$\triangleright\,${\cmntfont#1}$\,\triangleleft$}
%\def\SHC#1{\5\5\quad$\diamond\,${\cmntfont#1}}
\def\D{\defin{\#define}} % macro definition
\let\E=\equiv % equivalence sign
\def\ET{ and~} % conjunction between two section numbers
\def\ETs{, and~} % conjunction between the last two of several section numbers
\def\F{\defin{format}} % format definition
\let\G=\ge % greater than or equal sign
% \H is long Hungarian umlaut accent
\let\I=\ne % unequal sign
\def\J{\.{@\&}} % TANGLE's join operation
\let\K== % assignment operator
%\let\K=\leftarrow % "honest" alternative to standard assignment operator
```

```
% \L is Polish letter suppressed-L
\outer\def\M#1{\MN{#1}\ifon\vfil\penalty-100\vfilneg % beginning of section
  \vskip\intersecskip\startsection\ignorespaces}
\outer\def\N#1#2#3.{\gdepth=#1\gtitle={#3}\MN{#2}% beginning of starred section
  \ifon\ifnum#1<\secpagedepth \vfil\eject % force page break if depth is small
    \else\vfil\penalty-100\vfilneg\vskip\intersecskip\fi\fi
  \message{*\secno} % progress report
  \edef\next{\write\cont{\ZZ{#3}{#1}{\secno}% write to contents file
                  {\noexpand\the\pageno}}}\next % \ZZ{title}{depth}{sec}{page}
  \ifpdf\special{pdf: outline #1 << /Title (#3) /Dest
    [ @thispage /FitH @ypos ] >>}\fi
  \ifon\startsection{\bf#3.\quad}\ignorespaces}
\def\MN#1{\par % common code for \M, \N
  {\xdef\secstar{#1}\let\*=\empty\xdef\secno{#1}}% remove \* from section name
  \ifx\secno\secstar \onmaybe \else\ontrue \fi
  \mark{{{\tensy x}\secno}{\the\gdepth}{\the\gtitle}}}
% each \mark is {section reference or null}{depth plus 1}{group title}
% \O is Scandinavian letter O-with-slash
% \P is paragraph sign
\def\Q{\note{This code is cited in section}} % xref for mention of a section
\def\Qs{\note{This code is cited in sections}} % xref for mentions of a section
\let\R=\lnot % logical not
% \S is section sign
\def\T#1{\leavevmode % octal, hex or decimal constant
  \hbox{$\def\?{\kern.2em}%
    \def\$##1{\egroup_{\,\rm##1}\bgroup}% suffix to constant
    \def\_{\cdot 10^{\aftergroup}}% power of ten (via dirty trick)
    \let\~=\oct \let\^=\hex {#1}$}}
\def\U{\note{This code is used in section}} % xref for use of a section
\def\Us{\note{This code is used in sections}} % xref for uses of a section
\let\V=\lor % logical or
\let\W=\land % logical and
\def\X#1:#2\X{\ifmmode\gdef\XX{\null$\null}\else\gdef\XX{}\fi %$% section name
  \XX$\langle\,${\let\I=\ne#2\eightrm\kern.5em
    \ifacro{\pdfnote#1.}\else#1\fi}$\,\rangle$\XX}
\def\Y{\par\yskip}
\let\Z=\le
\let\ZZ=\let % now you can \write the control sequence \ZZ
\let\*=*

\let\Xand=\W
\def\Xandxeq{\MRL{{\W}{\K}}}
\let\Xbitand=\AND
\let\Xbitor=\OR
\let\Xcompl=\CM
\let\Xnot=\R
\let\Xnotxeq=\I
\let\Xor=\V
\def\Xorxeq{\MRL{{\OR}{\K}}}
\let\Xxor=\XOR
\def\Xxorxeq{\MRL{{\XOR}{\K}}}

%\def\oct{\hbox{\rm\char'23\kern-.2em\it\aftergroup\?\aftergroup}} % WEB style
%\def\hex{\hbox{\rm\char"7D\tt\aftergroup}} % WEB style
\def\oct{\hbox{$^\circ$\kern-.1em\it\aftergroup\?\aftergroup}}% CWEB style
\def\hex{\hbox{$^{\scriptscriptstyle\#}$\tt\aftergroup}} % CWEB style
\def\vb#1{\leavevmode\hbox{\kern2pt\vrule\vtop{\vbox{\hrule
        \hbox{\strut\kern2pt\.{#1}\kern2pt}}
      \hrule}\vrule\kern2pt}} % verbatim string

\def\onmaybe{\let\ifon=\maybe} \let\maybe=\iftrue
\newif\ifon \newif\iftitle \newif\ifpagesaved

\def\lheader{\mainfont\the\pageno\eightrm\qquad\grouptitle\hfill\title\qquad
  \mainfont\topsecno} % top line on left-hand pages
\def\rheader{\mainfont\topsecno\eightrm\qquad\title\hfill\grouptitle
  \qquad\mainfont\the\pageno} % top line on right-hand pages
\def\grouptitle{\let\i=I\let\j=J\uppercase\expandafter{\expandafter
                        \takethree\topmark}}
\def\topsecno{\expandafter\takeone\topmark}
\def\takeone#1#2#3{#1}
```

```
\def\taketwo#1#2#3{#2}
\def\takethree#1#2#3{#3}
\def\nullsec{\eightrm\kern-2em} % the \kern-2em cancels \qquad in headers

\let\page=\pagebody \raggedbottom
% \def\page{\box255 }\normalbottom % faster, but loses plain TeX footnotes
\def\normaloutput#1#2#3{\ifodd\pageno\hoffset=\pageshift\fi
 \shipout\vbox{
  \vbox to\fullpageheight{
  \iftitle\global\titlefalse
  \else\hbox to\pagewidth{\vbox to10pt{}\ifodd\pageno #3\else#2\fi}\fi
  \vfill#1}} % parameter #1 is the page itself
  \global\advance\pageno by1}

\gtitle={\.{CWEB} output} % this running head is reset by starred sections
\mark{\noexpand\nullsec0{\the\gtitle}}
\def\title{\expandafter\uppercase\expandafter{\jobname}}
\def\topofcontents{\centerline{\titlefont\title}\vskip.7in
  \vfill} % this material will start the table of contents page
\def\startpdf{\ifpdftex\pdfcatalog{/PageMode /UseOutlines}\else
    \ifpdf{\special{pdf: docview << /PageMode /UseOutlines >>}}\fi\fi}
\def\botofcontents{\vfill
  \centerline{\covernote}} % this material will end the table of contents page
\def\covernote{}
\def\contentspagenumber{0} % default page number for table of contents
\newdimen\pagewidth \pagewidth=6.5in % the width of each page
\newdimen\pageheight \pageheight=8.7in % the height of each page
\newdimen\fullpageheight \fullpageheight=9in % page height including headlines
\newdimen\pageshift \pageshift=0in % shift righthand pages wrt lefthand ones
\def\magnify#1{\mag=#1\pagewidth=6.5truein\pageheight=8.7truein
  \fullpageheight=9truein\setpage}
\def\setpage{\hsize\pagewidth\vsize\pageheight} % use after changing page size
\def\contentsfile{\jobname.toc} % file that gets table of contents info
\def\readcontents{\input \contentsfile}
\def\readindex{\input \jobname.idx}
\def\readsections{\input \jobname.scn}

\newwrite\cont
\output{\setbox0=\page % the first page is garbage
  \openout\cont=\contentsfile
       \write\cont{\catcode `\noexpand\@=11\relax}   % \makeatletter
  \global\output{\normaloutput\page\lheader\rheader}}
\setpage
\vbox to \vsize{} % the first \topmark won't be null

\def\ch{\note{The following sections were changed by the change file:}
  \let\*=\relax}
\newbox\sbox % saved box preceding the index
\newbox\lbox % lefthand column in the index
\def\inx{\par\vskip6pt plus 1fil % we are beginning the index
  \def\page{\box255 } \normalbottom
  \write\cont{} % ensure that the contents file isn't empty
       \write\cont{\catcode `\noexpand\@=12\relax}   % \makeatother
  \closeout\cont % the contents information has been fully gathered
  \output{\ifpagesaved\normaloutput{\box\sbox}\lheader\rheader\fi
    \global\setbox\sbox=\page \global\pagesavedtrue}
  \pagesavedfalse \eject % eject the page-so-far and predecessors
  \setbox\sbox\vbox{\unvbox\sbox} % take it out of its box
  \vsize=\pageheight \advance\vsize by -\ht\sbox % the remaining height
  \hsize=.5\pagewidth \advance\hsize by -10pt
    % column width for the index (20pt between cols)
  \parfillskip 0pt plus .6\hsize % try to avoid almost empty lines
  \def\lr{L} % this tells whether the left or right column is next
  \output{\if L\lr\global\setbox\lbox=\page \gdef\lr{R}
    \else\normaloutput{\vbox to\pageheight{\box\sbox\vss
        \hbox to\pagewidth{\box\lbox\hfil\page}}}\lheader\rheader
    \global\vsize\pageheight\gdef\lr{L}\global\pagesavedfalse\fi}
  \message{Index:}
  \parskip 0pt plus .5pt
  \outer\def\I##1, ##2.{\par\hangindent2em\noindent##1:\kern1em
    \ifacro\pdfnote##2.\else##2\fi.} % index entry
```

```
  \def\[##1]{$\underline{##1}$} % underlined index item
  \rm \rightskip0pt plus 2.5em \tolerance 10000 \let\*=\lapstar
  \hyphenpenalty 10000 \parindent0pt
  \readindex}
\def\fin{\par\vfill\eject % this is done when we are ending the index
  \ifpagesaved\null\vfill\eject\fi % output a null index column
  \if L\lr\else\null\vfill\eject\fi % finish the current page
  \parfillskip 0pt plus 1fil
  \def\grouptitle{NAMES OF THE SECTIONS}
  \ifacro \def\outsecname{Names of the sections} \let\Xpdf\X \fi
  \ifpdftex \makebookmarks \pdfdest name {NOS} fitb
    \pdfoutline goto name {NOS} count -\secno {\outsecname}
    \def\X##1:##2\X{\Xpdf##1:##2\X
      \firstsecno##1.\toks0{##2}\pdfoutline goto num \the\toksA {\the\toks0}}
  \else\ifpdf
    \special{pdf: outline -1 << /Title (\outsecname)
      /Dest [ @thispage /FitH @ypos ] >>}
    \def\X##1:##2\X{\Xpdf##1:##2\X
      \firstsecno##1.\toks0{##2} \special{pdf: outline 0 << /Title
        (\the\toks0) /A << /S /GoTo /D (\romannumeral\the\toksA) >> >>}}
  \fi\fi
  \let\topsecno=\nullsec
  \message{Section names:}
  \output={\normaloutput\page\lheader\rheader}
  \setpage
  \def\note##1##2.{\quad{\eightrm##1~\ifacro{\pdfnote##2.}\else{##2}\fi.}}
  \def\Q{\note{Cited in section}} % crossref for mention of a section
  \def\Qs{\note{Cited in sections}} % crossref for mentions of a section
  \def\U{\note{Used in section}} % crossref for use of a section
  \def\Us{\note{Used in sections}} % crossref for uses of a section
  \def\I{\par\hangindent 2em}\let\*=*
  \readsections}
\def\makebookmarks{% read contents info for PDF outlines (twice)
  \let\ZZ=\scanbookmarkline \readcontents\relax
  \let\ZZ=\writebookmarkline \readcontents\relax}
\def\expnumber#1{\expandafter\ifx\csname#1\endcsname\relax 0%
  \else \csname#1\endcsname \fi} % Petr Olsak's macros from texinfo.tex
\def\advancenumber#1{\countA=\expnumber{#1}\relax \advance\countA by1
  \expandafter\xdef\csname#1\endcsname{\the\countA}}
\def\scanbookmarkline#1#2#3#4{% remember last level item and add to parent
  \expandafter\xdef\csname curr#2\endcsname{#3}
  \ifnum#2>0\countB=#2 \advance\countB by-1
    \advancenumber{chunk\the\countB.\expnumber{curr\the\countB}}\fi}
\def\writebookmarkline#1#2#3#4{\pdfoutline goto num #3
  count -\expnumber{chunk#2.#3}{#1}}
\def\con{\par\vfill\eject % finish the section names
% \ifodd\pageno\else\titletrue\null\vfill\eject\fi % for duplex printers
  \rightskip 0pt \hyphenpenalty 50 \tolerance 200
  \setpage \output={\normaloutput\page\lheader\rheader}
  \titletrue % prepare to output the table of contents
  \pageno=\contentspagenumber
  \def\grouptitle{TABLE OF CONTENTS}
  \message{Table of contents:}
  \topofcontents \startpdf
  \line{\hfil Section\hbox to3em{\hss Page}}
  \let\ZZ=\contentsline
  \readcontents\relax % read the contents info
  \botofcontents \end} % print the contents page(s) and terminate
\def\contentsline#1#2#3#4{\ifnum#2=0 \smallbreak\fi
    \line{\consetup{#2}#1
      \rm\leaders\hbox to .5em{.\hfil}\hfil
      \ \ifacro\pdflink{#3}{\romannumeral#3}\else#3\fi\hbox to3em{\hss#4}}}
\def\consetup#1{\ifcase#1 \bf % depth -1 (@**)
  \or % depth 0 (@*)
  \or \hskip2em % depth 1 (@*1)
  \or \hskip4em \or \hskip6em \or \hskip8em \or \hskip10em % depth 2,3,4,5
  \else \hskip12em \fi} % depth 6 or more
\def\noinx{\let\inx=\end} % no indexes or table of contents
\def\nosecs{\let\FIN=\fin \def\fin{\let\parfillskip=\end \FIN}}
    % no index of section names or table of contents
\def\nocon{\let\con=\end} % no table of contents
```

```
\def\today{\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space\number\day, \number\year}
\newcount\twodigits
\def\hours{\twodigits=\time \divide\twodigits by 60 \printtwodigits
  \multiply\twodigits by-60 \advance\twodigits by\time :\printtwodigits}
\def\gobbleone1{}
\def\printtwodigits{\advance\twodigits100
  \expandafter\gobbleone\number\twodigits
  \advance\twodigits-100 }
\def\TeX{{\ifmmode\it\fi
   \leavevmode\hbox{T\kern-.1667em\lower.424ex\hbox{E}\hskip-.125em X}}}
\def\,{\relax\ifmmode\mskip\thinmuskip\else\thinspace\fi}
\def\datethis{\def\startsection{\leftline{\sc\today\ at \hours}\bigskip
  \let\startsection=\stsec\stsec}}
  % say `\datethis' in limbo, to get your listing timestamped before section 1
\def\datecontentspage{%
  \def\topofcontents{\leftline{\sc\today\ at \hours}\bigskip
   \centerline{\titlefont\title}\vfill}} % timestamps the contents page
```

Apéndice C: Como usar macros CWEB

Las macros en `cwebmac` hacen posible producir una variedad de formatos sin editar la salida de `CWEAVE` y el propósito de este apéndice es para explicar unas de las posibilidades.

1. Cuatro fuentes han sido declaradas además de las fuentes estándares del formato `PLAIN`: Puedes decir '`{\mc UNIX}`' para obtener UNIX en mayúsculas de tamaño mediano; puedes decir '`{\sc STUFF}`' para obtener STUFF en mayúsculas pequeñas; y puedes seleccionar las fuentes más largas aún `\titlefont` y `\ttitlefont` en el título de tu documento, donde `\ttitlefont` es un estilo de tipo de letra de máquina de escribir. Hay macros `\UNIX/` y `\CEE/` para referir a UNIX y C con mayúsculas con tamaños medianos.

2. Cuando mencionas un identificador en texto TEX, normalmente lo llamas '`|identifier|`'. Pero también puedes decir '`\\{identifier}`'. La salida lucirá igual en ambos casos, pero la segunda alternativa no pone el *identificador* dentro del índice, ya que evita la traducción de `CWEAVE` desde el modo C. En el segundo caso deberás poner una barra diagonal invertida antes de cada carácter subrayado en el identificador.

3. Para obtener estilo parecido al tipo de letra de máquina de escribir, tal como cuando se referencia a '`CWEB`', puedes usar la macro '`\.`' (p. ej., '`\.{CWEB}`'). En el argumento de esta macro deberás insertar una barra diagonal invertida adicional antes de los símbolos listados como 'cadena de caracteres especiales' ('*special string characters*') en el índice de `CWEAVE`, i.e., antes de las barras diagonales invertidas y signos de dólar y similares. Un '`\␣`' acá resultará en el símbolo invisible de espacio en blanco; para obtener un espacio invisible después de una secuencia de control debes decir '`{␣}`'. Si la cadena de caracteres es larga, la puedes cortar en una sub-cadena de caracteres que sean separadas por '`\)`'; el último caso da una barra diagonal invertida discrecional si TEX tiene que cortar una línea aquí.

4. Las tres secuencias de control `\pagewidth`, `\pageheight`, y `\fullpageheight` pueden ser redefinidas en la sección del limbo al principio de tu archivo `CWEB`, para cambiar las dimensiones de cada página. La configuración por omisión

```
\pagewidth=6.5in
\pageheight=8.7in
\fullpageheight=9in
```

fueron usados para preparar este manual; `\fullpageheight` es `\pageheight` más espacio para el título y número de página en el tope de cada página. Si cambias cualesquiera de estas cantidades, deberás llamar a la macro `\setpage` inmediatamente después de hacer el cambio.

5. La macro `\pageshift` define una cantidad a través de que las páginas de mano derecha (i.e., páginas de números impares) son corridas a la derecha respecto a las de la mano izquierda (números pares). Al ajustar esta cantidad será posible obtener la salida de doble lado en la cual los números de página se alinean en los lados opuestos de cada hoja.

6. La macro `\title` aparecerá en el tope de cada página en mayúsculas pequeñas; es el nombre del trabajo a menos que sea redefinido.

7. La primera página usualmente es asignada como la página número 1. Para empezar en la página 16, con contenidos en la página 15, digamos: '`\def\contentspagenumber{15} \pageno=\contentspagenumber \advance\pageno by 1`'.

8. La macro `\iftitle` suprimirá la línea de cabecera (*header line*) si es definida con '`\titletrue`'. El valor normal es `\titlefalse` excepto por la tabla de contenidos; así, la página de contenidos es usualmente no enumerada.

Dos macros son provistas para dar flexibilidad a la tabla de contenidos: `\topofcontents` es invocada justo antes de la información de contenido es leída y `\botofcontents` es invocada justo después. Acá hay una definición típica:

```
\def\topofcontents{\null\vfill
  \titlefalse % include headline on the contents page
  \def\rheader{\mainfont The {\tt CWEAVE} processor\hfil}
  \centerline{\titlefont The {\ttitlefont CWEAVE} processor}
  \vskip 15pt \centerline{(Version 3.64)} \vfill}
```

Redefiniendo `\rheader` (cual es el título para las páginas de mano derecha) es suficiente en este caso poner la información deseada en el tope de la página de contenidos.

9. Los datos para la tabla de contenidos son escritos en un archivo que es leído después que a los índices se les ha aplicado TEX; hay una línea de datos para cada sección estrellada. El archivo `common.toc` podrá lucir como esto:

```
\ZZ {Introduction}{0}{1}{28}{}
\ZZ {The character set}{2}{5}{29}{}
```

así sucesivamente. La macro `\topofcontents` podría redefinirse `\ZZ` de tal manera que la información aparezca en cualquier formato deseado. (Véase también punto 19 más abajo.)

10. Algunas veces es necesario o deseable dividir la salida de `CWEAVE` en sub-archivos que puedan ser procesados separadamente. Por ejemplo, el listado de TEX se extiende a más de 500 páginas y eso es bastante para exceder la capacidad de muchos dispositivos de impresión y/o sus aplicaciones computacionales. Cuando un trabajo extremadamente largo no es cortado en pedazos más pequeños, el proceso entero podría ser corrompido por un simple error de algún tipo, siendo necesario comenzar todo de nuevo.

Aquí está una forma segura de quebrar el archivo tejido (*woven file*) en tres partes: Digamos que las piezas son α, β y γ, donde cada pieza empieza con una sección estrellada. Todas las macros deberán ser definidas en la sección de limbo de apertura de α y copias de este código TEX deberá ser puesto al principio de β y de γ. Para procesar las partes separadamente, necesitamos preocuparnos de dos cosas: Los números de la página de inicio de β y γ necesitan ser configuradas correctamente, y los datos de la tabla de contenidos de las tres ejecuciones necesitan ser acumuladas.

Las macros `cwebmac` incluyen dos secuencias de control `\contentsfile` y `\readcontents` que facilitan el proceso necesario. Incluimos '`\def\contentsfile{cont1}`' en la sección del limbo de α e incluimos '`\def\contentsfile{cont2}`' en la sección del limbo de β; esto causa que el TEX escriba los datos de contenido de α y β dentro de `cont1.tex` y `cont2.tex`. Ahora en γ decimos

```
\def\readcontents{\input cont1 \input cont2 \input \contentsfile};
```

Esto trae los datos de las tres piezas, en el orden correspondiente.

Sin embargo, aún necesitamos resolver el problema de la enumeración de las páginas. Una forma para hacerlo es incluir lo siguiente en el material del limbo para β:

```
\message{Please type the last page number of part 1: }
\read -1 to \temp \pageno=\temp \advance\pageno by 1
```

Entonces simplemente provees los datos necesarios para cuando TEX lo solicite; una construcción similar es usada al principio de γ.

Este método puede, por supuesto, ser usado para dividir un archivo tejido en varias piezas.

11. Algunas veces es agradable incluir cosas en el índice que son compuestas en una forma especial. Por ejemplo, podríamos querer tener una entrada de índice para 'TEX'. `CWEAVE` provee dos simples formas para componer una entrada de índice (a menos que la entrada es un identificador o una palabra reservada): '`@^`' entrega tipo de letra romana y '`@.`' tipo de letra de máquina de escribir. Pero si intentamos componer 'TEX' en tipo de letra romana diciendo, p. ej., '`@^\TeX@>`', el carácter de barra diagonal invertida se pone entremedio, y esta entrada no aparecería en el índice con varias letras T.

La solución es usar la característica '`@:`', declarar una macro que simplemente remueve una llave de ordenamiento como sigue:

```
\def\9#1{}
```

Ahora puedes decir, p. ej.., '`@:TeX}{\TeX@>`' en tu archivo `CWEB`; `CWEAVE` lo pone dentro del índice alfabéticamente, basado en la llave de ordenamiento y produce la llamada macro '`\9{TeX}{\TeX}`' la cual asegurará que la llave de ordenamiento no sea impresa.

Una idea similar puede ser usada para insertar material escondido dentro de nombres de sección de tal manera que están alfabetizadas de la forma que sea que pudieses querer. Algunas personas llaman a estos trucos como "refinamientos especiales" ("*special refinements*"); otros los llaman como "*kludges*." ("solución no inteligente a un problema.")

12. La secuencia de control `\secno` es configurada con el número de la sección que está siendo compuesta tipográficamente.

13. Si quieres listar solamente las secciones que han cambiado, juntas con el índice, pon el comando '`\let\maybe=\iffalse`' en la sección de limbo antes de la primera sección de tu archivo `CWEB`. Es habitual hacer esto como el primer cambio en tu archivo de cambios.

Esta característica tiene una limitación TEXnica, sin embargo: no puedes usarla en conjunto con las secuencias de control como `\proclaim` o `\+` o `\newcount` que el TEX ha declarado ser '`\outer`', porque TEX rechaza saltar silenciosamente tales secuencias de control. Una forma de evitar esta limitación, es decir,

```
\fi \let\proclaim\relax \def\proclaim{...} \ifon
```

donde `\proclaim` es redefinido para ser el mismo de siempre, pero sin una clasificación `\outer`. (`\fi` acá detiene el salto condicional y `\ifon` activa de nuevo.) Asimismo,

```
\fi \newcount\n \ifon
```

es una forma segura de usar `\newcount`. TEX plano ya provee una macro no de exterior `\tabalign` que hace el trabajo de `\+`; puedes decir

```
\fi \let\+\tabalign \ifon
```

si prefieres la anotación más corta `\+`.

14. Para tener salidas diferentes al inglés, redefine las macros `\A`, `\As`, `\ATH`, `\ET`, `\ETs`, `\Q`, `\Qs`, `\U`, `\Us`, `\ch`, `\fin`, `\con`, `\today`, `\datethis` y `\datecontentspage`. `CWEAVE` no necesita ser cambiado.

15. Alguna salida puede ser suprimida selectivamente con las macros `\noatl`, `\noinx`, `\nosecs`, `\nocon`.

16. Todos los acentos y símbolos de texto especiales del formato de TEX plano funcionarán en documentos `CWEB` como están explicados en el Capítulo 9 de *The TEXbook*, con una excepción: el acento de punto (normalmente `\.`) deberá ser escrito `\:` en vez de los tradicionales.

17. Varias líneas comentadas en `cwebmac.tex` son sugerencias que los usuarios podrán desear adoptar. por ejemplo, tal línea inserta una página en blanco si tienes una impresora duplex. Apéndices D, E y F de la versión completa de este manual son imprimidos usando la opción comentada (*commented-out*) que substituye '←' por '=' en los listados del programa. Al mirar a estos apéndices podría ayudar decidir cual formato te gusta más.

18. Andreas Scherer ha contribuido con una macro llamada `\pdfURL` con la cual puedes decir lo siguiente, en cualquier lugar de las partes TEX o de los comentarios C de un archivo `CWEB`:

```
You can send email to \pdfURL{the author}{mailto:andreas.scherer@@pobox.com}
or visit \pdfURL{his home page}{http://www.pobox.com/\TILDE/scherer}.
```

En un documento PDF, el primer argumento aparecerá en azul como texto posible de pinchar; el Acrobat reader, si es configurado correctamente, entonces redireccionará esos vínculos al navegador del usuario y abrirá o el cliente de email o el visualizador de HTML. En un documento impreso, ambos argumentos serán impresos (`el segundo entre paréntesis y letra de máquina de escribir`). Ciertos caracteres especiales en una dirección de Internet necesitan ser manejadas en una forma extraña, de tal manera `CWEAVE` y/o TEX no los confundirán con controles de formateo: Usa `@@` para `@` y `\TILDE/` para `~` y `\UNDER/` para `_`.

19. Documentos PDF contienen marcadores (*bookmarks*) que listan todos los títulos del grupo mayor en la tabla de contenidos, algunos de los cuales serán subsidiarios de otros si la característica de profundidad de `@*` ha sido usada. Tales entradas de marcadores son también conocidas como "bosquejos." ("*outlines.*") Por otra parte, el título de grupo final, 'Nombres de las secciones', pueden ser abiertas para listar cada nombre de sección; usuarios de Acrobat pueden por lo tanto, navegar más fácilmente en cualquier sección deseada.

Las macros de `cwebmac.tex` son prudentes de "desinfectar" ("*sanitize*") todos los nombres que aparecen como marcadores, al remover caracteres especiales y códigos de formateo que son inapropiados para las capacidades tipográficas limitadas de los bosquejos de PDF. Por ejemplo, una sección de `CWEAVE` es llamada 'Cases for *case_like*', la cual es representada por el código TEX '`Cases for \PB{\\{case\_like}}`' en `cweave.tex`; su nombre desinfectado es simplemente '`Cases for case_like`'. (Cuando los archivos `.pdf` son producidos, el quinto parámetro de cada `\ZZ` en el archivo `.toc` es configurado como la forma desinfectada del primer parámetro; ve el punto 9 arriba y el punto 20 abajo.)

En general, la desinfección remueve secuencias de control TEX y llaves, excepto por las secuencias de control definidas por `CWEB`. Tal traducción funciona la mayoría de las veces, pero puedes sobre-escribir las por omisión y puedes obtener cualquier traducción que desees al usar trucos TEXnicos. Por ejemplo, después

```
\sanitizecommand\foo{bar}
```

la secuencia de control `\foo` lo desinfectará a '`bar`'. Y después

```
\def\kluj#1\\{foo}
```

el código TEX '`\kluj bar\\`' se imprimirá como 'foo' pero desinfecta a '`bar`', porque las secuencias de control `\kluj` y `\\` son removidos por desinfección.

20. Además, los títulos de grupo pueden ser convertidos en un texto arbitrario desinfectado mientras que también al cambiar sus formas al ejecutar los títulos, al usar `\ifheader`. Considera, por ejemplo, un archivo fuente `CWEB` que comience con las dos líneas

```
\def\klujj#1\\{\ifheader FOO\else foo\fi}
@*Chinese \klujj bar\.
```

Esta codificación presenta un grupo mayor titulado '**Chinese foo**', al ejecutar el título 'CHINESE FOO' y la entrada de tabla de contenidos 'Chinese foo'. El marcador correspondiente es, sin embargo, '`Chinese bar`'. y la entrada de archivo correspondiente `.toc` es '`\ZZ {Chinese \klujj bar\\}{1}{1}{1}{Chinese bar}`'.